MASTER IN COMPUTER SCIENCE

Sécurisation de la notification de présence dans le protocole SIP à l'aide de la carte d'identité électronique

Gamby, Sébastien

Award date:
2006

Facultés Universitaires Notre-Dame de la Paix, Namur
Institut d'informatique
Année académique 2005-2006

Sécurisation de la notification de présence dans le protocole SIP à l'aide de la carte d'identité électronique

–

Sébastien Gamby

Mémoire présenté en vue de l'obtention du grade de licencié en informatique.

Résumé

Ce travail porte sur la sécurisation de la notification de présence du protocole SIP à l'aide de la carte d'identité électronique. Autrement dit, sur l'authentification de l'utilisateur auprès du serveur proxy lors de l'envoi de requêtes REGISTER à l'aide de la carte d'identité électronique de ce dernier. La solution qui a été développée pour permettre cette sécurisation a été d'ajouter aux requêteq REGISTER un Authenticated Identity Body (AIB) pour permettre l'authentification de l'utilisateur tout en protégeant les en-têtes des requêtes d'une modification par un pirate afin d'usurper l'identité de l'utilisateur.

Abstract

This work is related to the securization of presence notification of SIP protocol thanks to the electronic identity card, more specifically, the authentication of the user to the proxy server during the sending of REGISTER requests thanks to the electronic identity card of this one. The solution that has been developped consists in adding to the REGISTER requests an Authenticated Identity Body (AIB) to enable the authentication of the user and protect the headers of the requests from any modification aimed at impersonating the real user.

Je souhaite remercier les professeurs Laurent Schumacher et Jean Ramaekers pour leurs conseils avisés tout au long de ce mémoire et je souhaite égalemment exprimer ma gratitude à ma famille pour le soutien qu'elle m'a apporté au cours de mes études.

Table des matières

Table des figures

Liste des acronymes

AIB	Authenticated Identity Body
AODF	Authenticated Object Directory File
API	Application Programming Interface
CA	Certificate Authority
CDF	Certificate Directory File
Cert	Certificate
CRL	Certificate Revocation List
CVS	Concurrent Version System
eID	electronic IDentity
DF	Dedicated File
DHCP	Dynamic Host Configuration Protocol
EF	Elementary File
HTTP	HyperText Transfert Protocol
IP	Internet Protocol
JMF	Java Media Framework
MF	Master File
MIME	Multipurpose Internet Mail Extensions
NIST	National Institute for Standards and Technology
OCSP	Online Certificate Status Protocol
ODF	Object Directory File
PKCS	Public Key Cryptography Standards
PuK	Public Key
PrK	Private Key
RFC	Request For Comments
RSA	Rivest Shamir Adleman
SIP	Session Initiation Protocol
URI	Uniforme Resource Identifier
SAML	Security Assertion Markup Language
SSH	Secure SHell
SSL	Secure Socket Layer
TLS	Transport Layer Security
XML	eXtented Markup Language

Introduction

A l'origine, le réseau Internet ne permettait que la transmission de courriers électroniques et la consultation de pages Web. Aujourd'hui, les services disponibles sont aussi nombreux que variés. Il est désormais possible de commander des plats à livrer, d'acheter des articles à des magasins situés de l'autre côté de la planète et bien d'autres choses encore depuis un terminal disposant d'une connexion au Net.

Malheureusement, Internet n'est pas un paradis numérique et les personnes qui emploient les services en ligne s'exposent à certains risques. L'un de ces risques est l'usurpation d'identité. Il s'agit de personnes malintentionnées qui emploient des services en ligne en se faisant passer pour d'autres personnes. De cette façon, elles peuvent par exemple acheter des biens et des services en employant le compte bancaire de leur victime ou encore accéder à des informations confidentielles.

Etant donné que l'administration fédérale belge souhaitait offrir des services administratifs en ligne à ses citoyens, il était dès lors important qu'elle trouve un moyen sûr pour authentifier les utilisateurs de ces services afin que personne ne puisse usurper leur identité dans le cadre d'opérations administratives. Aussi a-t-elle décider de remplacer les cartes d'identité traditionnelles par des cartes d'identité électroniques qui permettrent à leur propriétaire de prouver leur identité aussi bien face à un agent de police que lors de l'accès à un service en ligne.

De par sa nature, la carte d'identité électronique ne se limite pas à sécuriser l'accès à l'administration en ligne mais peut être utilisée pour permettre une authentification fiable des utilisateurs au sein d'autres services Internet. L'un des services qui pourrait bénéficier de cette amélioration est le protocole SIP. Ce dernier pourrait dans un proche avenir éclipser les communications téléphoniques en permettant à deux utilisateurs ou plus de communiquer entre eux au moyen d'une session (messagerie instantanée, communication audio, communication audiovisuelle, etc.).

L'un des concepts clef du protocole SIP est celui de la notification de présence, c'est à dire informer les utilisateurs sur les personnes qui sont actuellement connectées au service SIP ainsi que l'adresse IP où elles sont disponibles. L'objet de ce mémoire a été de sécuriser cette notification de présence à l'aide de la carte d'identité électronique.

Le premier chapitre présente la carte eID et ses fonctionnalités. Le second effectue un rapide rappel du fonctionnement du protocole SIP et détaille l'algorithme d'authentification employé par défaut dans ce protocole : la Digest Authentication. Le troisième chapitre explique le travail d'implémentation qui a été réalisé pour combiner l'authentification par carte eID et le protocole SIP. Le quatrième et dernier chapitre présente les futures réalisations qui pourraient faire suite à ce travail. Enfin une conclusion est proposée et trois annexes sont fournies qui détaillent respectivement la première version du galop d'essai, le fonctionnement du programme JAIN SIP Presence Proxy et le fonctionnement du programme JAIN SIP Applet Phone.

Chapitre 1
Présentation de la carte eID

L'objet de ce chapitre est de présenter la carte d'identité électronique, les données qu'elle contient ainsi que ses fonctionnalités. Son contenu est basé sur la lecture de la documentation fournie dans le CDROM de l'eID Development Kit acquis par les FUNDP. Ce kit est disponible à l'eID shop (`http://www.eid-shop.be`) et il contient :

- Des spécimens de carte eID,
- un lecteur de carte,
- des librairies Java et .Net pour employer le lecteur,
- des exemples de code.

1.1 Contenu de la carte eID

1.1.1 Le contenu visible

FIG. 1.1 – Une carte d'identité électronique, le recto est à gauche et le verso à droite

Comme présenté à la figure 1.1, la carte d'identité électronique est une sorte de croisement entre la carte d'identité traditionnelle et la carte à puce bancaire. Elle est néanmoins un véritable pièce d'identité officielle et de ce fait les informations suivantes peuvent être lues à sa surface :

- Au recto :
  - Le nom de famille du propriétaire,
  - se(s) prénom(s),
  - son lieu de naissance,
  - sa date de naissance,
  - son sexe,
  - sa nationalité,
  - le numéro de sa carte,

  - la date de début de validité de la carte,
  - la date de fin de validité,
  - sa photo.
- Au verso :
  - Le numéro d'identification du registre national,
  - le lieu de délivrance,
  - un code alphanumérique.

1.1.2 Le contenu électronique

Le contenu de la puce électronique de la carte eID est présenté dans la documentation fournie avec l'eID Development Kit [12]. Une image issue de cette documentation est montrée à la figure 1.2 et montre une partie du contenu de la puce électronique de la carte d'identité. Celle-ci contient entre autres deux répertoires importants. Le premier, DF(ID), sert à contenir l'application d'identification qui fournit les informations concernant le propriétaire, le second, DF(BELPIC Application), contient une application respectant le standard PKCS#15 v1.1[1] [7]. Cette dernière sert à effectuer des opérations cryptographiques qui permettront au propriétaire de la carte eID de s'authentifier et de signer des documents électroniques mais pas de les (dé)crypter. De plus, de l'espace libre a été laissé afin de permettre l'ajout de nouvelles fonctions plus tard, probablement des fonctionnalités de cryptage et de décryptage.

L'application d'identification

Le répertoire de cette application contient toutes les informations visibles sur la carte eID à l'exception du code alphanumérique situé au verso. A cela s'ajoute l'adresse et des signatures électroniques sur l'adresse, la photo et le reste des informations d'identité. L'adresse apparaît uniquement dans le contenu de la puce électronique afin que le propriétaire puisse changer de domicile sans avoir à changer de carte d'identité. Lorsque quelqu'un déménage, il lui suffit de se rendre au guichet Population de sa nouvelle commune afin qu'un employé communal mette à jour sa carte eID avec sa nouvelle adresse. Malheureusement, il y a un inconvénient à cela : les policiers des autres pays ne sont pas équipés de lecteur de carte eID et donc les citoyens belges en voyage à l'étranger sont obligés de posséder en plus de leur carte eID un document papier délivré par leur commune qui contient leur adresse.

L'application PKCS#15 v1.1

Les données du répertoire de cette application respectent le standard PKCS#15 v1.1 développé par RSA Laboratories [7] et contient des clefs RSA privées et pu-

[1]Ce standard décrit les données que doit contenir une smartcard, c'est à dire une carte électronique servant à s'authentifier au moyen de clefs privées RSA associées à leurs certificats.

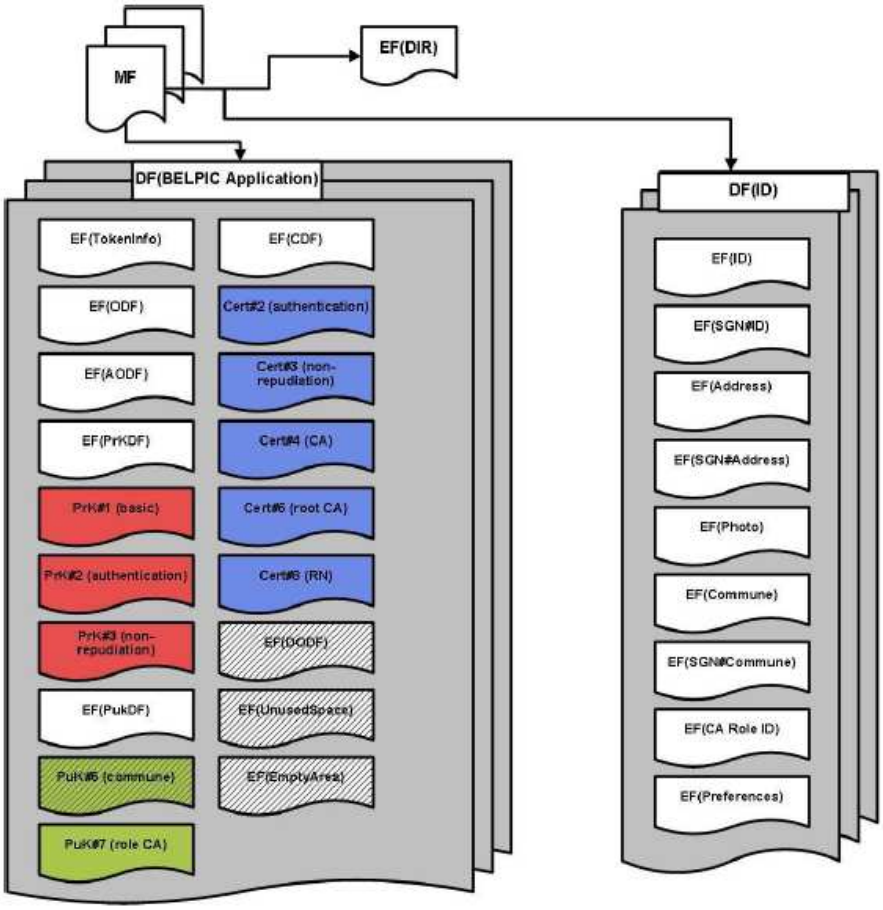

FIG. 1.2 – Illustration issue de [12] montrant le contenu des deux répertoires contenant les applications d'identification et de signature

bliques de 1024 bits ainsi que des certificats X.509. Le tableau de la figure 1.3 représente les éléments de ce fichier ainsi que les relations entre eux. Chaque ligne du tableau représente une fonctionnalité de la carte d'identité électronique. Les deux lignes qui ont vraiment de l'importance dans le cadre de ce mémoire sont les lignes *Authentication* et *Non-repudiation*. La ligne *Authentication* correspond à la fonction d'authentification et la ligne *Non-repudiation* correspond à la fonction de signature de documents électroniques. Deux autres éléments intéressants sont les certificats Cert#4 (CA) et Cert#6 (Root). Le certificat Cert#4 est celui dont la clef privée correspondante a servi à signer les certificats Cert#2 et Cert#3 tandis que le certificat Cert#6 est un certificat auto-signé dont la clef privée correspondante a servi à signer le certificat Cert#4.

Les autres éléments sont moins intéressants dans le cadre de ce mémoire. La clef privée PrK#1 permet au middleware de la carte eID de s'authentifier auprès d'une application. La clef PuK#7 permet à des logiciels de s'authentifier auprès du middleware de la carte. L'emploi de la clef PuK#5 a été abandonné. Enfin, le Cert#8 permet de vérifier la signature des données personnelles (toutes les informations

d'identité excepté l'adresse), celle de l'adresse et celle de la photo.

	Private Key (Java Object)	Public Key (Java Object)	X.509 Certificates (Transparent file)
Basic	PrK#1		
Authentication	PrK#2	In Cert#2	Cert#2
Non-repudiation	PrK#3	In Cert#3	Cert#3
Certification Authority (CA)		In Cert#4	Cert#4
Commune		PuK#5	
Root			Cert#6
CA Role		PuK#7	
RN			Cert#8

FIG. 1.3 – Les clefs et les certificats contenus dans le fichier PKCS#15 de la carte eID avec leurs relations (figure issue de [12]).

1.1.3 Accès aux données électroniques

L'un des documents fournis dans le CDROM de l'eID Development Kit [8] décrit le détail des commandes qu'un logiciel peut passer à une carte d'identité électronique à l'aide d'un lecteur de carte adapté afin d'activer les fonctions de la carte. En se basant sur ces spécifications, un logiciel peut accéder aux données suivantes :

- La totalité des informations d'identité sur le propriétaire,
- le certificat Cert#2,
- le certificat Cert#3,
- le certificat Cert#4,
- le certificat Cert#6,
- le certificat Cert#8.

Le middleware de la carte ne permet pas de lire d'autres informations pour des raisons de sécurité. De plus, seules des applications disposant de la clef privée PrK#7 peuvent éditer le contenu de la carte après s'être authentifiées auprès de celle-ci avec cette clef.

En plus d'accéder à ces informations, un logiciel peut accéder à certaines fonctions du middleware de la carte eID. Parmi celles-ci les plus intéressantes sont :

- La vérification du code PIN,
- l'initialisation d'une signature électronique,
- la signature électronique.

1.2 Fonctionnalités de sécurité de la carte eID

A présent que les informations et les fonctionnalités importantes de la carte eID sont connues, l'emploi de la carte eID comme outil de sécurisation électronique va être développé.

1.2.1 L'authentification et la signature de documents électroniques

Les fonctionnalités d'authentification et de signature de documents électroniques font toutes deux intervenir le même mécanisme : celui de la signature électronique. Pour rappel, une signature électronique se fait sur un morceau de données numériques et s'obtient en faisant subir à ce dernier un hashage suivi d'un cryptage à l'aide d'une clef assymétrique privée (voir figure 1.4). La vérification de la signature se fait en décryptant celle-ci avec la clef publique correspondant à la clef privée qui a servi au cryptage et en vérifiant que le résultat est bien équivalent au hashage du morceau de données initial (voir figure 1.5).

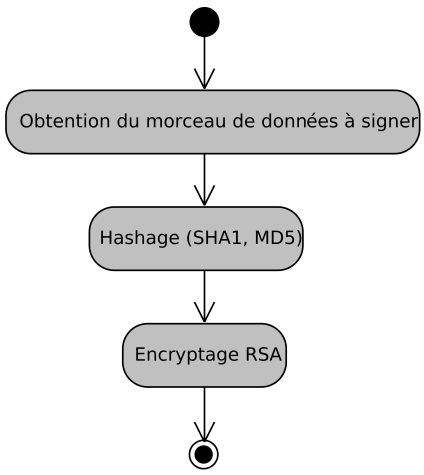

FIG. 1.4 – Diagramme d'activité représentant la création d'une signature électronique

La carte d'identité électronique permet d'effectuer seulement la seconde phase d'une signature électronique : le cryptage du hashage. Il faut donc que le logiciel utilisant la carte eID pour produire une signature électronique effectue lui même le hashage du morceau de données à signer avant de le transmettre à la carte qui effectuera le cryptage à l'aide d'une de ses clefs privées (PrK#2 ou PrK#3).

La procédure à suivre pour effectuer le cryptage est représentée à la figure 1.6 :

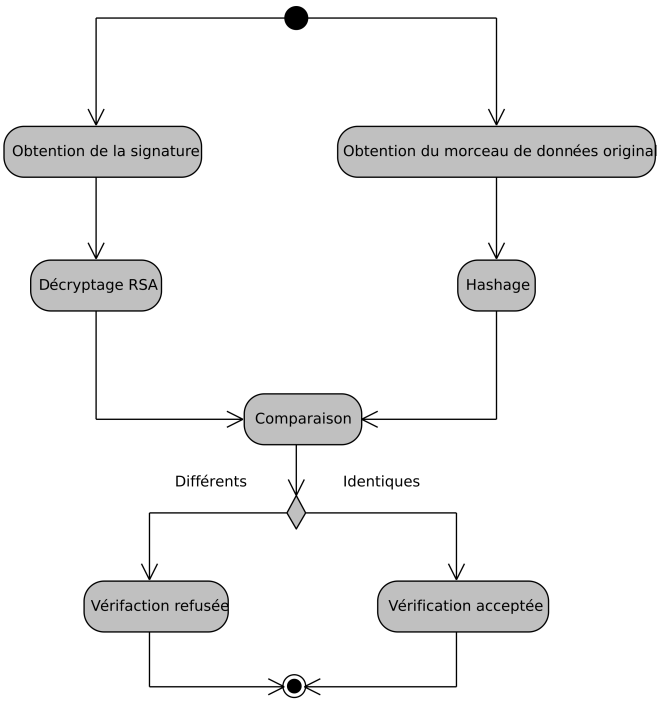

FIG. 1.5 – Diagramme d'activité représentant la vérification d'une signature électronique

1. Initialiser la signature en précisant l'algorithme de hashage (SHA-1, MD5 ou autre)[2] et la clef privée qui va être utilisée (PrK#2 ou PrK#3) ;
2. vérifier le code PIN ;
3. effectuer la signature en fournissant le morceau de données hashé selon l'algorithme spécifié précédemment.

La différence qu'il y a à employer la clef privée PrK#2 ou la clef PrK#3 est que dans le premier cas, une seule vérification du code PIN suffit pour un nombre illimité de signatures électroniques et que dans le second cas, il faut effectuer une vérification du code PIN avant chaque signature.

Pour effectuer une authentification, le logiciel employant la carte eID doit réaliser une signature avec la clef privée PrK#2 sur un challenge (un morceau de données généré aléatoirement) fourni par l'entité exigeant l'authentification, tandis que pour une signature électronique il faut employer la clef privée PrK#3 pour crypter un hashage du document électronique à signer.

[2]Chaque algorithme de hashage produit une chaîne de octets de longueur fixe, 16 octets pour MD5, 20 octets pour SHA-1. Spécifier l'algorithme permet donc de vérifier la validité du hashage et d'annuler la procédure de signature si la longueur de la chaîne issue du hashage est incompatible avec l'algorithme spécifié. Si l'algorithme n'est pas spécifié, la chaîne doit avoir une longueur inférieur au maximum de octets que peut traiter la carte : 117.

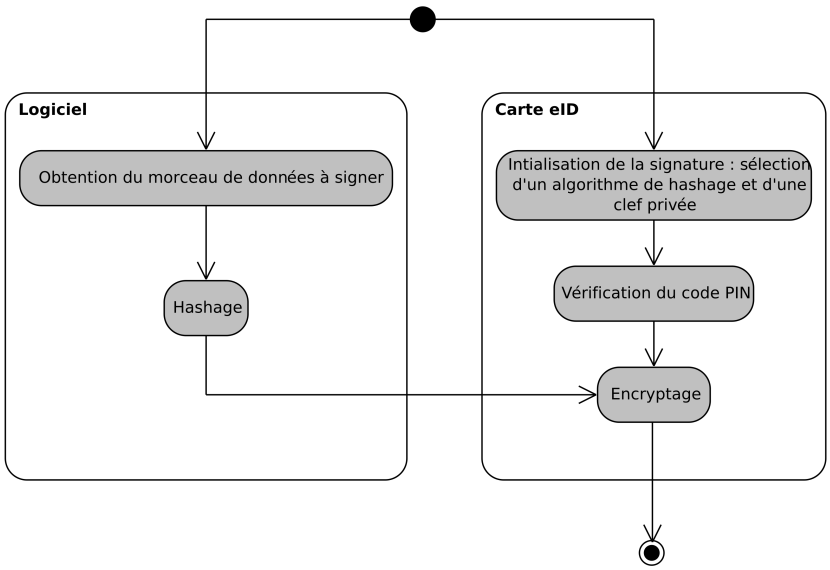

FIG. 1.6 – Diagramme d'activité représentant la création d'une signature électronique avec une carte d'identité électronique

1.2.2 Fonctionnalités de sécurité extérieures à la carte

L'administration fédérale belge fournit deux services de sécurité qui permettent d'augmenter la fiabilité de la carte d'identité électronique. Il s'agit de la publication de CRL et d'un serveur OCSP.

CRL est l'acronyme pour Certificate Revocation List, il s'agit donc de listes contenant les numéros de série des certificats qui pour une raison quelconque (décès, vol de la carte, péremption, etc.) sont révoqués par l'administration fédérale belge. Un logiciel qui souhaite employer ce service doit télécharger le dernier CRL en date sur le site `http://crl.eid.belgium.be` et vérifier que le numéro de certificat avec lequel le propriétaire d'une carte eID souhaite s'authentifier ne figure pas sur cette liste. Si c'est le cas, alors le logiciel doit refuser l'authentification même si la signature effectuée avec la carte est valide.

OCSP est l'acronyme de Online Certificate Status Protocol, c'est donc un protocole permettant de vérifier en ligne le statut d'un certificat . Ce protocole est présenté dans la RFC 2560 [9]. Le logiciel qui souhaite utiliser ce service doit se connecter au serveur situé à l'adresse `ocsp.eid.belgium.be` et employer ce protocole. Grâce à ce service, un état est associé au certificat. Cet état peut être bon, inconnu ou révoqué. Si l'état est différent de bon, le logiciel doit refuser l'authentification même si la signature produite par la carte est valide.

1.3 “Le galop d’essai”

Ce mémoire a commencé par un “galop d’essai”. Il s’agissait de réaliser une petite application employant l’authentification eID afin de se familiariser avec cette dernière. Dans un premier temps le galop d’essai consistait à intégrer l’authentification eID au protocole SSH mais ce projet a été abandonné, les raisons en sont données dans l’annexe de ce travail. Le galop d’essai a donc consisté à développer un petite application client-serveur où le serveur authentifie le client grâce à sa carte eID.

Le fonctionnement du programme est montré à la figure 1.7. L’utilisateur lance l’application client qui contacte le serveur, celui-ci renvoie un challenge que l’application client va signer avec la clef privée PrK#2 de la carte eID et le code PIN obtenu auprès de l’utilisateur. Ensuite l’application client envoie au serveur le challenge signé et le certificat Cert#2. Le serveur vérifie la validité de la signature et le statut du certificat grâce au serveur OCSP ; si les deux vérifications sont effectuées avec succès l’utilisateur est authentifié. Enfin, le serveur envoie à l’application client le résultat de l’authentification et celle-ci répercute l’information à l’utilisateur.

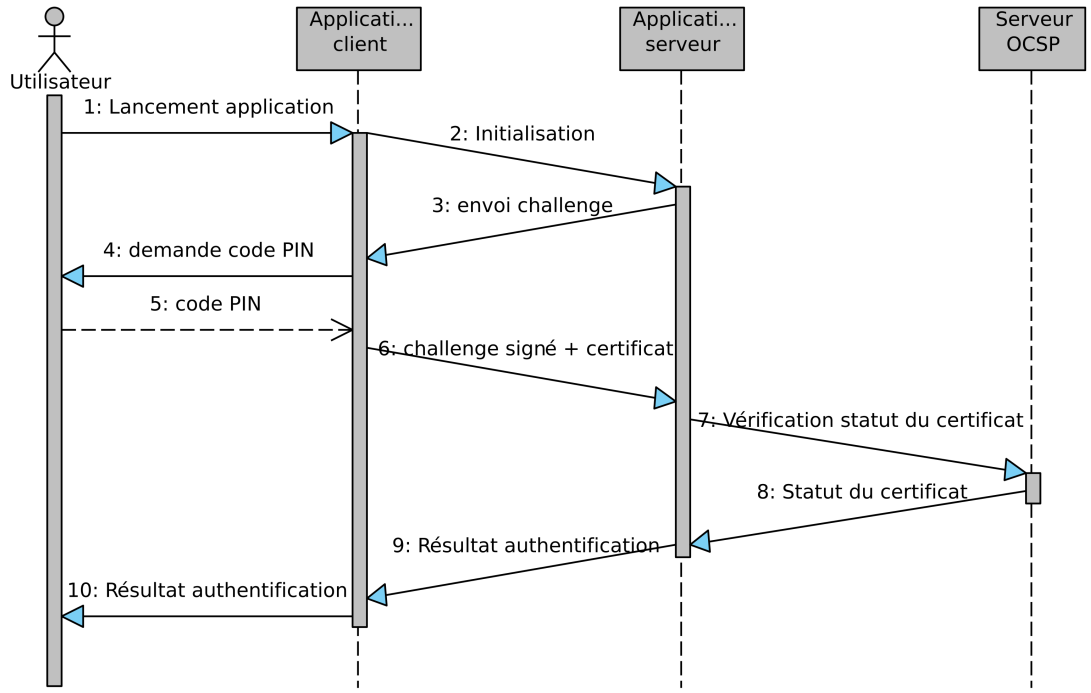

Fig. 1.7 – Diagramme de séquence montrant l’échange des messages entre les composants de l’application du galop d’essai.

Cette application a été réalisée en Java en utilisant l’API *be.certipost.app.card* fournie dans le CDROM de l’eID Development Kit qui permet d’accéder aux données et fonctionnalités de la carte eID, et l’API *JPC/SC 0.80* disponible sur le site `http://www.linuxnet.com` qui permet de piloter les lecteurs de carte eID.

Chapitre 2
Authentification de base de SIP

Le but de ce chapitre est de présenter le mécanisme d'authentification de base du protocole SIP. Pour ce faire, il commence par un court rappel du but et du fonctionnement du protocole SIP avant d'aborder la Digest Authentication que le protocole SIP emploie par défaut.

2.1 Rappel sur le protocole SIP

La plupart des terminaux capables de communiquer grâce à Internet (PC, PDA, notebooks, etc.) ne sont pas connectés en permanence et emploient le protocole DHCP pour obtenir une adresse IP. De plus, un bon nombre de ces terminaux sont "nomades", c'est à dire qu'ils voyagent et se connectent depuis un grand nombre d'endroits différents. Enfin, beaucoup d'entre eux sont employés par plusieurs utilisateurs et logiciels différents ce qui pose la question de savoir qui est en train d'utiliser quelle machine à quel moment. Le principal problème que rencontre donc un individu ou un logiciel désireux d'en contacter un autre est de savoir quelle adresse IP il doit employer pour contacter celui-là à un moment donné.

Le protocole SIP décrit dans la RFC 3261 [11] permet de résoudre ce problème en offrant cinq services à ses utilisateurs :

- La localisation des correspondants (trouver leur adresse IP),
- leur disponibilité (connecté, hors ligne, etc.),
- leurs capacités techniques (les protocoles de communication gérés par la machine qu'ils utilisent),
- l'initialisation d'une session de communication,
- l'administration des sessions.

Ce protocole utilise trois principaux types d'intervenants :

- User agent (le terminal que l'utilisateur emploie),
- serveur proxy,
- serveur registrar.

Le fonctionnement (simplifié) est présenté à la figure 2.8 : lorsqu'un utilisateur (humain ou logiciel) désire en contacter un autre, il envoie une requête de type INVITE contenant l'URI SIP du destinataire à un des serveurs proxy de son domaine Internet. Si l'utilisateur à contacter (le destinataire) appartient au même domaine, le proxy traite directement la requête, sinon il la transfère à un proxy du domaine du destinataire. Le proxy en charge du traitement de la requête retrouve l'adresse IP du destinataire grâce à un "location service" constitué par une base de données contenant des correspondances logiques entre URI SIP et adresse IP, ces correspondances sont appellées "bindings". Une fois l'adresse IP du destinataire connue, le

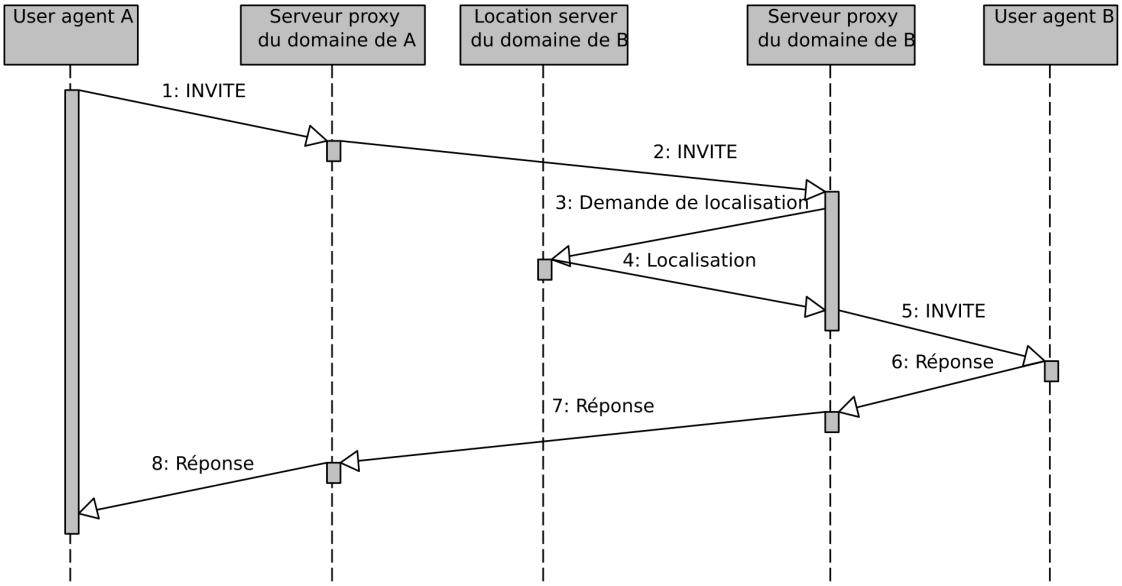

FIG. 2.8 – Sequence diagram montrant l'acheminement d'une requête INVITE à destination

proxy transfère la requête au destinataire. La réponse fait le même parcours mais en sens inverse et sans appel au location service.

Le location service est donc l'épine dorsale du protocole SIP puisque c'est lui qui permet de retrouver l'adresse IP à laquelle le destinataire d'une requête peut être joint. Le développeur du service SIP peut choisir lui-même le moyen d'implémentation de ce service, mais généralement ce service est assuré par le serveur registrar.

Le serveur registrar a pour but de traiter les requêtes de type REGISTER que les utilisateurs lui envoient. Ces requêtes servent à permettre la notification de présence, c'est à dire permettre aux utilisateurs d'informer le service SIP de l'adresse IP qu'ils emploient actuellement et où ils peuvent donc être joints. Un exemple de requête de ce type est présenté à la figure 2.9. Elles ont pour particularité d'avoir un corps vide et donc de n'être composées que d'en-têtes. Parmi ceux-ci, From et Contact sont particulièrement intéressants puisque le premier contient l'URI SIP de l'utilisateur (sip :sgamby@fundp.ac.be dans cet exemple) et le second contient son adresse IP (192.168.1.3 dans cet exemple). En traitant ces requêtes REGISTER, le serveur registrar pourra donc générer un binding qu'il va mémoriser si c'est lui qui implémente le location service ou le transférer à l'entité réseau en charge du location service (le location server) dans le cas contraire.

Il reste deux détails importants à signaler avant de conclure cette section. D'abord, un seul et même terminal peut assurer les rôles du serveur registrar et du serveur proxy. Ensuite, les requêtes REGISTER peuvent ne pas être envoyées par les utilisateurs directement au serveur registrar mais transiter par le serveur proxy qui dès lors s'occupe de les relayer au serveur registrar.

REGISTER sip :localhost :4000 SIP/2.0
From : <sip :sgamby@fundp.ac.be> ;tag=1186
To : <sip :sgamby@fundp.ac.be>
Contact : <sip :sgamby@192.168.1.3 :7772 ;transport=udp>
Call-ID : e0409e887146b3e29085dfc6da03e2dc@192.168.1.3
Cseq : 2 Register

empty body

FIG. 2.9 – Exemple de requête REGISTER

2.2 Digest Authentication

Il est recommandé de sécuriser la notification de présence sans quoi une personne malintentionnée pourrait s'enregister auprès du service SIP avec l'URI d'une autre personne et donc communiquer avec des tiers en usurpant l'identité du propriétaire de cette URI. Cette personne malintentionnée pourrait par exemple appeler le banquier de sa victime pour transférer de l'argent, effectuer des actes illégaux (arnaques téléphoniques par exemple), etc. Pour éviter cela, il faut sécuriser la notification de présence en authentifiant l'utilisateur lorsqu'il envoie une requête REGISTER au serveur registrar.

L'algorithme d'authentification entre l'utilisateur et le serveur proxy spécifié dans la RFC de SIP [11] pour les requêtes de tous types est la Digest Authentication qui a été développée initialement pour le protocole HTTP et est décrite plus en détail dans la RFC 2617 [4]. Ce mécanisme est basé sur l'emploi d'un nom d'utilisateur et d'un mot de passe qui sont connus de l'utilisateur et du serveur proxy. Ce dernier doit avoir en mémoire le nom d'utilisateur et le mot de passe correspondant de chacune des personnes autorisées à employer le service SIP. En général, le couple nom d'utilisateur - mot de passe est généré lors de l'inscription d'un nouvel utilisateur auprès du service SIP et mémorisé à ce moment là. L'échange des messages entre l'user agent et le serveur proxy au cours de l'envoi d'une requête quelconque avec authentification de l'utilisateur est présenté à la figure 2.10.

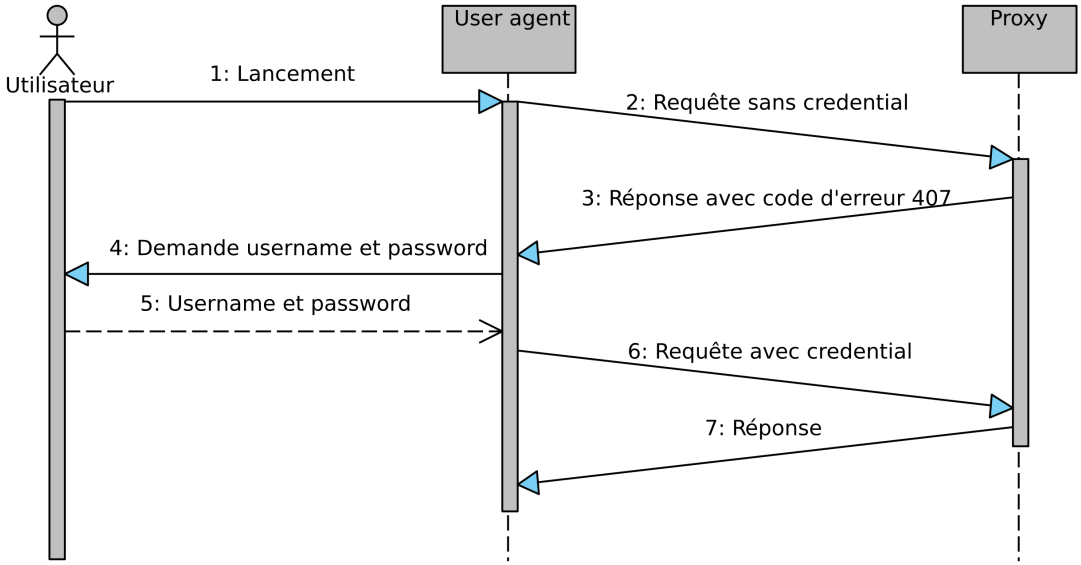

FIG. 2.10 – Diagramme de séquence montrant l'échange des messages entre l'user agent et le proxy lors d'une Digest Authentication

En général, l'user agent envoie une première requête sans credential (données assurant l'authentification). Le serveur envoie une réponse au user agent avec le code d'erreur 407 "Proxy Authentication Required", cette réponse est la copie de la requête à laquelle il a ajouté un en-tête Proxy-Authenticate contenant entre autres un challenge. Une fois cette réponse reçue, l'user agent demande à l'utisateur son nom et son mot de passe. Il emploie un algorithme de hashage (MD5 par défaut) pour générer une réponse au challenge à partir de celui-ci, du nom de l'utilisateur et du mot de passe. L'user agent renvoie la requête en remplaçant l'en-tête Proxy-Authenticate par un en-tête Proxy-Authorization qui contient notamment le nom de l'utilisateur, le challenge et la réponse au challenge et qui sert de credential. Le serveur proxy authentifie l'utilisateur en retrouvant dans sa base de données le mot de passe correspondant au nom d'utilisateur, en répétant l'opération de hashage et en vérifiant que le résultat correspond bien à la réponse de l'user agent au challenge (voir figure 2.11).

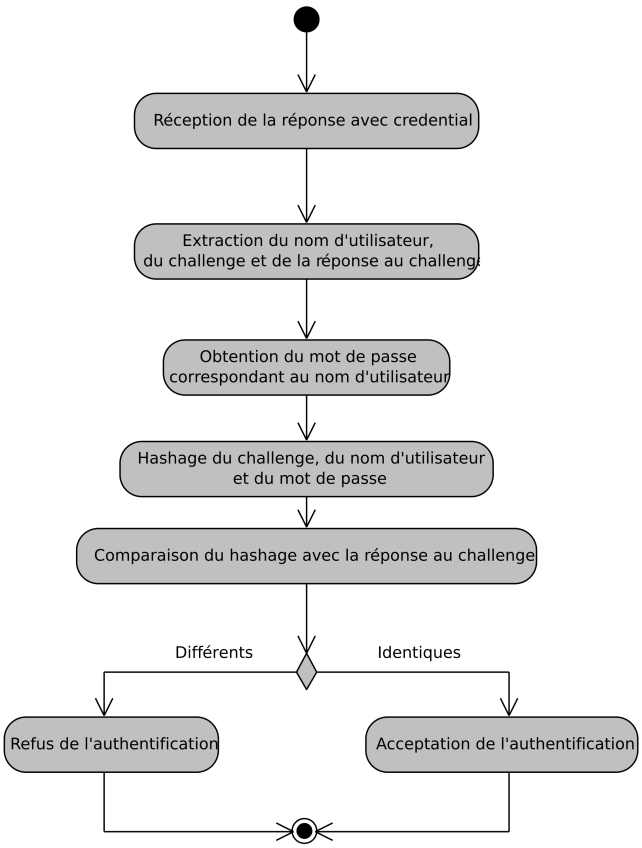

FIG. 2.11 – Diagramme d'activité montrant la vérification d'une Digest Authentication.

Le pilier de la Digest Authentication est donc le partage d'un secret entre le serveur proxy et chacun des utilisateurs, lequel consiste en un couple nom d'utilisateur - mot de passe. Grâce à ce système, il est possible d'authentifier l'utilisateur auprès du serveur proxy lors de l'envoi d'une requête REGISTER et donc de sécuriser la notification de présence. Cependant, si quelqu'un parvient à obtenir le couple nom d'utilisateur - mot de passe d'une autre personne, par exemple en installant un logiciel espion sur l'ordinateur de celle-ci, la fiabilité de cet algorithme d'authentification sera compromise. Il serait donc plus fiable d'employer un algorithme d'authentification basé sur l'emploi de la carte d'identité électronique puisqu'il faudrait impérativement posséder la carte pour pouvoir s'authentifier.

Chapitre 3
Réalisations

Ce chapitre présente le travail d'implémentation qui a été réalisé durant ce mémoire dans le but d'employer l'authentification eID pour sécuriser la notification de présence sous SIP. Il est important de signaler que ce travail ne couvre que l'authentification d'un utilisateur et donc la protection contre l'usurpation d'identité. La protection contre d'autres risques, notamment l'espionnage, n'est pas abordée.

3.1 Choix des programmes

Développer entièrement une application SIP avec un user agent, un serveur proxy et un serveur registar implémentant le location service n'était pas réalisable étant donné le volume des spécifications du protocole et les moyens disponibles pour ce mémoire. De ce fait, il a été décidé de récupérer des programmes existants et de les modifier afin de remplacer la Digest Authentication implémentée par défaut par un algorithme d'authentification employant la carte eID.

Des recherches ont donc été effectuées afin de trouver des programmes implémentant les différentes entités d'un service SIP et qui soient en licence libre afin d'obtenir facilement leur code source. La première piste étudiée a été celle des applications softphone fournies avec les distributions Linux (Kphone, Linphone, Gnomemeeting, etc.). Ces applications implémentent l'user agent mais elles n'ont pas été retenues car elles sont écrites en C ou C++ et que les librairies fournies avec l'eID Development Kit sont elles écrites en Java ou en .Net. Le même problème s'est posé avec la plupart des applications implémentant les serveurs proxy et registrar.

Les programmes qui ont été finalement choisis sont : JAIN-SIP Applet Phone et JAIN-SIP Presence Proxy. Il s'agit tous deux de programmes Java développés par le National Institute of Standards and Technology (NIST) en licence libre. Il est également à noter que les deux programmes sont toujours en cours de développement mais que les algorithmes d'enregistrement et d'authentification sont déjà implémentés. De plus, ils sont tous les deux basés sur l'API JAIN-SIP qui implémente la création et la gestion de requêtes et de réponses SIP. Le code source de ces programmes peut être téléchargé grâce à CVS après s'être enregistré comme observateur du projet sur le site `https://voip.dev.java.net`.

Le programme JAIN-SIP Applet Phone est une applet qui peut également être employée comme un programme ordinaire. Il implémente un user agent permettant de communiquer par messagerie instantanée et par communication audio (real time ou voice messaging). Il emploie l'API Java Media Framework (JMF) qui doit avoir été installé sur la machine utilisée. Cet API est disponible à l'adresse `http://java.sun.com/products/java-media/jmf/`.

Le programme JAIN-SIP Presence Proxy quant à lui implémente le serveur proxy

et peut implémenter le serveur registar avec le location service (configurable). Dans le cadre de ce travail, une seule instance de ce programme a été employée et elle implémentait donc les deux types de serveur.

Ces deux programmes ont donc été retenus pour être modifiés afin d'intégrer l'emploi de la carte d'identité électronique pour sécuriser la notification de présence. Les raisons qui ont principalement motivé leur choix en dehors du fait qu'ils faisaient partie des rares programmes implémentant des entités SIP et écrits en Java sont que premièrement leur code source dispose d'une documentation adéquate tant pour la Javadoc que pour les commentaires, et ensuite que des explications sur l'utilisation des programmes et notament sur leurs fichiers de configuration sont disponibles sur le site `http://snad.ncslm.nist.gov/proj/iptel/`.

3.2 Etude de spécifications techniques sur l'authentification

Une fois les programmes choisis, il a fallu développer un algorithme qui permettrait de sécuriser la notification de présence avec la carte d'identité électronique. Pour être plus précis, il fallait développer un algorithme qui permettrait d'authentifier l'utilisateur auprès du serveur proxy lors du traitement d'une requête REGISTER. De plus cette authentification devait être basée sur la vérification d'une signature électronique effectuée à l'aide de la carte eID. Dans ce but, des recherchers ont été effectuées dans la documentation technique disponible en ligne afin de trouver des spécifications existantes pour le protocole SIP ou compatibles avec celui-ci, décrivant des algorithmes d'authentifications de l'utilisateur auprès d'une entité réseau au moyen d'une signature électronique.

Les spécifications de ce type qui ont été examinées au cours de ce travail sont :

- SAML [2],
- P-Asserted_Identity [5],
- XML-Signature [1],
- AIB [10].

Chacune d'entre elles va maintenant être présentée et critiquée.

3.2.1 SAML

Cette spécification est basée sur l'emploi du language XML et permet à une autorité SAML (une entité du système considérée comme fiable) de faire des assertions de sécurité sur un utilisateur. Un exemple de ce genre d'assertion est présenté à la figure 3.12. Ces assertions de sécurité peuvent concerner l'identité de l'utilisateur, ses attributs et son niveau d'autorisation.

L'emploi du protocole SAML pour authentifier les utilisateurs dans un service Internet nécessite donc l'intégration d'au moins une autorité SAML à ce service, c'est-à-dire un serveur auprès duquel les utilisateurs doivent s'authentifier par un moyen supposé fiable afin d'obtenir une assertion SAML qui contient des informations d'authentification avec lesquelles ils pourront prouver leur identité auprès des autres serveurs. La documentation de SAML précise que le choix du moyen d'authentification entre l'autorité SAML et les utilisateurs est laissé à la discrétion de l'implémenteur mais elle définit malgré tout un standard XML pour des requêtes d'authentification envoyées par les utilisateurs à l'autorité SAML et les réponses renvoyées par celle-ci qui contiennent une assertion SAML en cas de succès de l'authentification. La fiabilité d'une requête d'authentification peut être assurée par l'insertion dans cette dernière d'une signature électronique effectuée sur elle-même avec une clef privée dont l'autorité SAML possède la clef publique correspondante laquelle peut être contenue dans un certificat. De plus, chaque requête doit impérativement contenir la date de sa création avec une précision qui va jusqu'à la seconde. Il est donc difficile de rejouer une requête si l'autorité SAML effectue une vérification sur cette date de façon à ne pas tolérer un délai trop long entre son émission et son traitement.

Les assertions SAML sont la base de ce protocole et il faut donc assurer leur fiabilité en rendant impossible ou presque leur contrefaçon par des pirates. Pour cela, il est possible aux autorité SAML de signer leurs assertions avec une clef privée. Bien sûr, chacun des serveurs du service Internet susceptibles de recevoir des requêtes contenant une assertion SAML signée doit posséder une copie de la clef publique ou du certificat correspondant à la clef privée de l'autorité SAML. La fiabilité des assertions SAML peut encore être accrue en leur adjoignant une période de validité suffisamment courte pour leur éviter d'être ré-utilisées ainsi que l'adresse IP dont les requêtes les contenant doivent être originaires pour être valables afin d'empêcher la modification de l'origine des requêtes.

Comme les requêtes d'authentification du protocole SAML peuvent contenir une signature électronique, il est possible de combiner ce protocole avec l'usage de la carte d'identité électronique en employant cette dernière pour effectuer les signatures des requêtes d'authentification. Pour permettre la vérification des signatures, il faut soit qu'une copie des certificats Cert#2 soit conservée chez l'autorité SAML, soit que l'utilisateur fournisse ce certificat avec sa signature et qu'une vérification de la validité de celui-ci soit effectuée avec le protocole OCSP. Le détail des messages échangés entre les entités d'un service SIP employant le protocole SAML et la carte eID dans le cadre de l'envoi d'une requête de l'user agent de l'utilisateur au serveur

proxy est présenté à la figure 3.13. L'user agent envoie une requête d'authentification signée avec la carte eID de l'utilisateur à l'autorité SAML qui après vérification de la signature eID lui envoie une réponse contenant une assertion SAML. L'user agent peut maintenant intégrer l'assertion SAML à une requête qu'il envoie au serveur proxy. Il reste à ce dernier à vérifier la validité de l'assertion SAML de la requête reçue, c'est-à-dire vérifier que la signature électronique effectuée par l'autorité SAML sur l'assertion est valide, que l'adresse IP contenue dans l'assertion est bien celle dont la requête est originaire et enfin que la période de validité de l'assertion n'est pas dépassée. Si toutes ces vérifications réussissent, le serveur proxy peut traiter la requête (dans le cas d'une requête REGISTER, il enregistre la présence de l'utilisateur à l'origine de la requête).

Pour résumer l'emploi du protocole SAML dans un service Internet quelconque a les avantages suivants :

- Les assertions SAML peuvent être signées (difficile à contrefaire),
- elles peuvent avoir une période de validité (difficile à rejouer),
- elles peuvent être associées à une adresse IP (difficile de modifier leur origine).

Ainsi que les inconvénients suivants :

- Il faut ajouter un nouveau serveur au service Internet (autorité SAML),
- il faut conserver une copie des certificats Cert#2 chez l'autorité SAML,
- la clef publique ou le certificat de l'autorité SAML doit être distribuée chez tous les autres serveurs.

```
<Assertion ID=''_a75adf55-01d7-40cc-dbd8372ebdfc"
    IssueInstant="2003-04-17T00 :46 :02Z" Version="2.0"
    xmlns="urn :oasis :names :tc :SAML :2.0 :assertion">
    <Issuer>https ://www.opensaml.org/IDP</Issuer>
    <Subject>
      <NameID
        Format="urn :oasis :names :tc :SAML :1.1 :nameid-format :
emailAddress">
        scott@example.org
      </NameID>
      <Subject
        Confirmation Method="urn :oasis :names :tc :SAML :2.0 :cm :bearer">
    </Subject>
    <Conditions NotBefore="2003-04-17T00 :46 :02Z"
      NotOnOrAfter="2003-04-17T00 :51 :02Z">
      <AudienceRestriction>
        <Audience>http ://www.opensaml.org/SP</Audience>
      </AudienceRestriction>
    </Conditions>
    <AuthnStatement AuthnInstant="2003-04-17T00 :46 :00Z">
      <AuthnContext>
        <AuthnContextClassRef>
          urn :oasis :names :tc :SAML :2.0 :ac :classes :Password
        </AuthnContextClassRef>
      </AuthnContext>
    </AuthnStatement>
</Assertion¿
```

FIG. 3.12 – Exemple d'assertion SAML délivrée par une autorité SAML

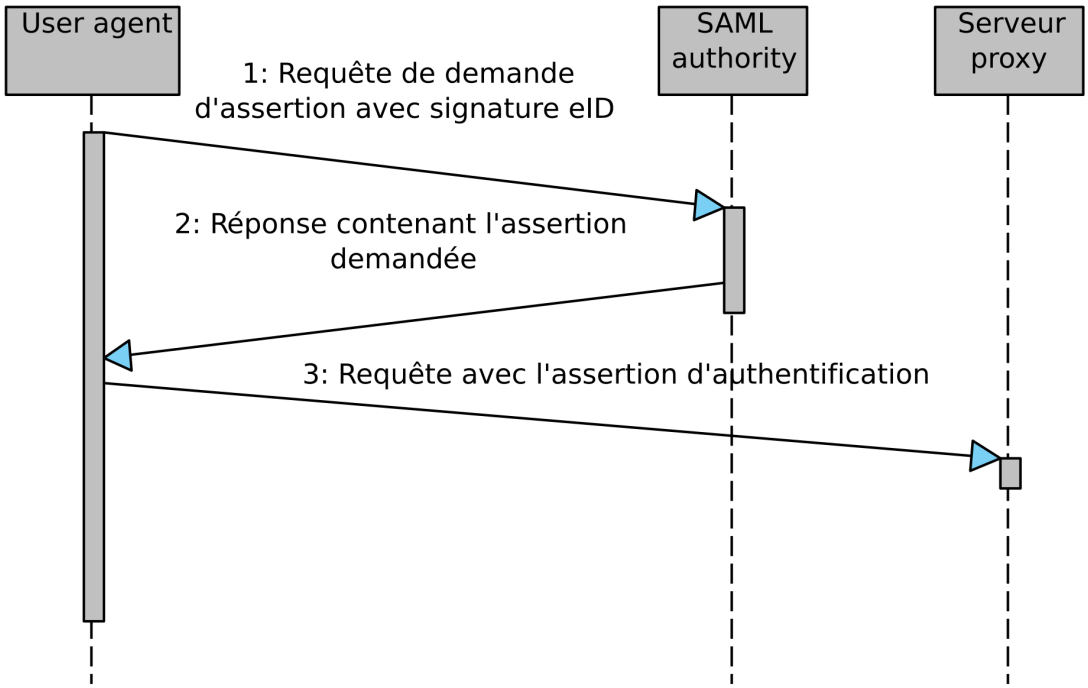

FIG. 3.13 – Sequence diagram représentant l'intégration d'une autorité SAML dans le protocole SIP

3.2.2 P-Asserted-Identity

Cette spécification ne concerne pas vraiment l'authentification proprement dite. Elle concerne seulement les grands systèmes SIP contenant plusieurs serveurs proxy qui forment un réseau par lequel transitent les requêtes. Le problème qui se pose est que chacun des serveurs proxy recevant une requête d'un utilisateur va demander une authentification de ce dernier l'obligeant ainsi à s'identifier auprès de chacun des serveurs proxy par lesquel transite sa requête. La solution proposée dans cette spécification est que l'utilisateur s'authentifie normalement auprès du premier proxy de la chaîne de transmission de sa requête et que celui-ci remplace le credential de la requête (un en-tête Proxy-Authorization par défaut) par un nouvel en-tête nommé P-Asserted-Identity qui contient le nom de l'utilisateur et son URI SIP et qui indique que l'utilisateur a déjà été authentifié. Evidemment, l'usage de ce protocole implique de sécuriser les communications entre les serveurs proxy pour éviter que quelqu'un ne court-circuite le processus d'authentification.

La figure 3.14 représente l'échange des messages entre les entités. L'user agent effectue une authentification avec le serveur proxy numéro 1, c'est à dire qu'il envoie une première requête sans credential, reçoit une réponse 407 "Proxy Authorization Required" contenant un challenge à signer et envoie enfin une requête avec un en-tête Proxy-Authorization contenant la signature électronique sur le challenge. Après

avoir vérifié la signature, le serveur proxy numéro 1 remplace cet en-tête par un en-tête P_Asserted_Identity. La requête ainsi modifiée est transférée de proche en proche dans le reste de la chaîne formée par les serveurs proxy jusqu'à sa destination.

En définitive, cette spécification a pour avantage d'éviter des authentifications en chaîne lors du transfert d'une requête SIP à travers plusieurs serveurs proxy. Son inconvénient est qu'elle exige le cryptage des communications entre les serveurs proxy pour contrer les éventuels abus de ce système.

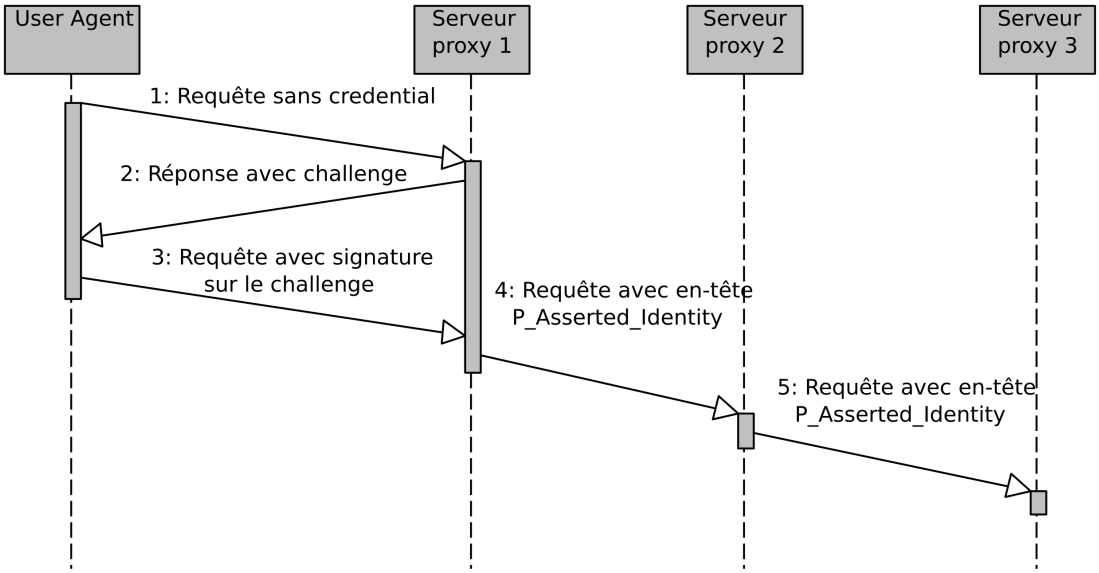

FIG. 3.14 – Sequence diagram représentant l'emploi d'en-tête P_Asserted_Identity dans le protocole SIP

3.2.3 XML-Signature

Cette spécification décrit le codage en language XML d'une signature électronique. Un exemple de ce type de signature est présentée à la figure 3.15. Pour permettre la vérification de ce genre de signature, deux solutions sont possibles : soit une copie des certificats ou des clefs publiques est conservée chez l'entité réseau auprès de laquelle les utilisateurs veulent s'authentifier, soit une copie du certificat est intégrée dans la signature XML et une vérification de ce dernier lors de la validation de la signature est effectuée.

Cette spécification peut être employée dans un service Internet quelconque pour authentifier les utilisateurs en permettant à ces derniers d'adjoindre au corps de leurs requêtes une signature électronique. Pour éviter que les requêtes soient rejouées, il faut que la valeur qui sert de support pour la signature électronique soit unique pour chaque requête (par exemple la date ou une valeur aléatoire).

Il est possible d'employer la carte eID pour générer des signatures XML et donc d'utiliser cette spécification pour intégrer l'usage de la carte d'identité électronique dans la sécurisation de la notification de présence dans le protocole SIP. Cependant, il est impossible de prévenir la modification des en-têtes des requêtes. Il est donc possible à un pirate d'intercepter une requête REGISTER avec une signature XML valide et de remplacer l'adresse IP de l'en-tête Contact de celle-ci par la sienne pour s'authentifier. En envoyant cette requête modifiée au serveur proxy, il peut ainsi s'enregistrer avec l'URI SIP de sa victime et son adresse IP.

En résumé, les avantages de cette spécifications sont :

- La possibilité d'intégrer une signature électronique dans le corps des requêtes,
- la possibilité d'ajouter les certificats aux signatures.

Les inconvénients quant à eux sont :

- L'impossibilité d'empêcher que les requêtes soient rejouées.

```
<Signature Id="MyFirstSignature" xmlns="http ://www.w3.org/2000/09/
xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod Algorithm="http ://www.w3.org/TR/2001/
REC-xml-c14n-20010315"/>
    <SignatureMethod Algorithm="http ://www.w3.org/2000/09/xmldsig#
dsa-sha1"/>
    <Reference URI="http ://www.w3.org/TR/2000/REC-xhtml1-200001
26/">
      <Transforms>
        <Transform Algorithm="http ://www.w3.org/TR/2001/REC-xml-
c14n-20010315"/>
      </Transforms>
      <DigestMethod Algorithm="http ://www.w3.org/2000/09/xmldsig#
sha1"/>
      <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
  <KeyInfo>
    <KeyValue>
      <DSAKeyValue>
        <P>...</P><Q>...</Q><G>...</G><Y>...</Y>
      </DSAKeyValue>
    </KeyValue>
  </KeyInfo>
</Signature>
```

FIG. 3.15 – Exemple de signature électronique encodée en XML

3.2.4 AIB

Les requêtes des protocoles “HTTP like”[3] sont transmises en clair sur le réseau. De ce fait, il est facile pour un pirate de les intercepter et de les modifier notamment de façon à usurper l'identité d'un utilisateur. Dans le cas particulier des requêtes du protocole SIP, un pirate pourrait remplacer l'adresse IP de l'en-tête Contact par la sienne de façon à s'enregister avec l'URI SIP de sa victime.

La spécification AIB a été développée pour lutter contre cette menace. Elle a été développée uniquement pour le protocole SIP, mais il est possible de l'appliquer à d'autres protocoles HTTP like. La solution qu'elle propose est d'ajouter au corps des requêtes un Authenticated Identity Body (AIB) qui contient une ou plusieurs signatures électroniques sur une sélection des en-têtes. Cela permet d'identifier un ou plusieurs utilisateurs grâce à leur signature électronique tout en protégeant l'intégrité des en-têtes des requêtes puisque le fait de modifier l'un de ceux qui appartiennent à la sélection signée entraîne l'échec de la vérification de la (les) signature(s). Un exemple de requête REGISTER SIP contenant un AIB est présenté à la figure 3.16. Un AIB est un objet MIME multipart/signed constitué de deux éléments : premièrement un message/sipfrag contenant la sélection des éléments qui sert de base à la signature électronique et ensuite un application/pkcs#7 qui contient un objet SignedData qui respecte le standard PKCS#7 [4] développé par les laboratoires RSA [6]. Ce dernier contient la signature électronique de chacun des utilisateurs à l'origine de la requête et le certificat correspondant.

En employant cette spécification dans le protocole SIP, il est possible d'employer la carte d'identité électronique pour sécuriser la notification de présence sous SIP. Il suffit que chaque signature électronique contenue dans un AIB soit réalisée avec une carte eID et que le certificat Cert#2 correspondant soit copié dans cet AIB.

En résumé, les avantages de la spécification AIB sont :

- Possibilité d'authentifier un ou plusieurs utilisateurs avec une signature électronique,
- protection de l'intégrité d'une sélection des en-têtes,
- le transport des certificats correspondants aux signatures.

Aucun inconvénient notable n'a été remarqué.

[3] Les protocoles dont les requêtes et les réponses respectent le modèle : en-tête, ligne vide, corps.

[4] Ce standard permet d'ajouter des signatures électroniques effectuées avec des clefs asymétriques RSA à des courriels.

REGISTER sip :localhost :4000 SIP/2.0
Call-ID : d1b5504b5a28427fc80c9c6c4a51d643@192.168.1.3
CSeq : 2 REGISTER
From : <sip :sgamby@fundp.ac.be> ;tag=3418
To : <sip :sgamby@fundp.ac.be>
Via : SIP/2.0/UDP 192.168.1.3 :3711 ;branch=z9hG4bKf446e8ced596b471e3a3062df
be63d8a
Max-Forwards : 2
Contact : <sip :sgamby@192.168.1.3 :3711 ;transport=udp>
Proxy-Authorization : AIB username="21267647932559286040145475065891120497"
,realm="fundp.ac.be",nonce="964c226416a4794644cb29c2a29ec683d715ad08"
Content-Type : multipart/signed
Content-Length : 2955

——=_Part_0_13177628.1152625817750

Content-Type : message/sipfrag
Content-Disposition : aib ; handling=optional

From : <sip :sgamby@fundp.ac.be> ;tag=3418
To : <sip :sgamby@fundp.ac.be>
Contact : <sip :sgamby@192.168.1.3 :3711 ;transport=udp>
Call-ID : d1b5504b5a28427fc80c9c6c4a51d643@192.168.1.3
CSeq : 2 REGISTER
Proxy-Authorization : AIB username="21267647932559286040145475065891120497"
,realm="fundp.ac.be",nonce="964c226416a4794644cb29c2a29ec683d715ad08"

——=_Part_0_13177628.1152625817750

Content-Type : application/pkcs7-signature ; name=smime.p7s
Content-Transfer-Encoding : base64
Content-Disposition : attachment ; filename=smime.p7s ;handling=required

SignedData PKCS#7 encodé en base 64

——=_Part_0_13177628.1152625817750–

FIG. 3.16 – Exemple de requête sécurisée avec un AIB.

3.3 Présentation des programmes avant modification

Cette section a pour but de présenter des notions sur le fonctionnement des deux programmes qui permettront une meilleure compréhension des modifications qui y ont été apportées pour intégrer l'usage de la carte d'identité électronique à la sécurisation de la notification de présence.

3.3.1 JAIN SIP Presence Proxy

Ce programme sert à implémenter le rôle du serveur proxy et optionnellement celui du serveur registrar. Son utilisation est présentée dans l'annexe B. Il emploie un fichier de configuration XML dont le chemin d'accès précédé de "-CF" doit être donné en argument lors du lancement du programme. Ce fichier peut être édité directement avec un programme de traitement de texte ou bien via le menu de configuration de l'application JAIN SIP Presence Proxy. Il sert à contenir différentes options de configurations dont les plus intéressantes dans le cadre de ce mémoire sont :

- L'activation ou la non-activation du service registrar,
- l'activation ou la non-activation de l'authentification des utilisateurs,
- la classe Java à employer pour exécuter l'authentification,
- le chemin du fichier contenant les informations nécessaires à l'authentification des utilisateurs (authentication file).

Dans le cadre de ce mémoire, le service registrar et l'authentification des utilisateurs a été activée. La classe Java qui implémente l'authentification peut être choisie grâce au fichier de configuration mais il doit impérativement implémenter une interface : AuthenticationMethod qui se trouve dans le paquetage gov.nist.sip.proxy.authentication. Le contenu de l'authentication file quant à lui est déterminer par la classe Java qui implémente l'authentification.

Lorsque le serveur reçoit une requête, il la valide avant de la traiter. Cette validation couvre plusieurs points dont l'authentification de l'utilisateur fait partie. Cette authentification est effectuée en créant une nouvelle instance de la classe implémentant l'authentification, en l'initialisant avec le chemin de l'authentication file et en lui fournissant la requête. Dans la version originelle, la classe employée est DigestServerAuthentication du paquetage gov.nist.sip.proxy.authentication. Celle-ci nécessite que son authentication file soit un fichier XML contenant les noms d'utilisateurs et les mots de passe correspondants. Elle effectue la vérification en se basant sur l'en-tête Proxy-Authorization. Elle en extrait le nom d'utilisateur dont elle retrouve le mot de passe correspondant dans l'authentication file, elle effectue ensuite le hashage requis pour une Digest Authentication et elle vérifie que le résultat obtenu correspond au contenu du champ response de cet en-tête. Si la validation (y compris l'authentification) est réussie, le programme traite la requête.

Pour intégrer l'emploi de la carte d'identité électronique pour sécuriser la notification de présence dans cette application, le plus simple est de créer une nouvelle classe implémentant l'interface AuthenticationMethod. Au lieu de vérifier une Digest Authentication celle-ci vérifiera une authentification effectuée avec un algorithme employant la carte eID. De plus, cette classe aura besoin d'un autre type d'authentication file.

3.3.2 JAIN SIP Applet Phone

Ce programme sert à implémenter l'user agent de l'utilisateur. Son fonctionnement est présenté dans l'annexe C. Contrairement à JAIN SIP Presence Proxy, il ne possède pas de fichier de configuration et il n'est donc pas possible de choisir la classe Java qui implémente l'authentification. Par conséquent, intégrer l'authentification des utilisateurs à l'aide de la carte d'identité électronique nécessite la ré-écriture de la classe implémentant l'authentification. Cette dernière est la classe DigestClientAuthenticationMethod qui implémente l'interface ClientAuthenticationMethod du même paquetage.

Lorsque ce programme envoie une requête au serveur proxy, il ne met pas de credential dans cette dernière par défaut. Si le serveur proxy exige une authentification, il renvoie une réponse Proxy Authorization Required. Lorsqu'il reçoit cette réponse, le programme crée une instance de la classe DigestClientAuthenticationMethod, il l'initialise avec notamment le nom d'utilisateur et le mot de passe qui ont été obtenus auprès de l'utilisateur ainsi que le challenge contenu dans l'en-tête Proxy-Authenticate de la réponse et ensuite il lui demande de générer un hashage Digest Authentication qu'il intègre dans le champ response de l'en-tête Proxy-Authorization de la requête qu'il renvoie au serveur proxy.

Intégrer l'usage de la carte eID nécessite de modifier la classe DigestClientAuthenticationMethod de façon à employer un algorithme d'authentification basé sur la carte d'identité électronique.

3.4 Première réalisation

Cette section présente la première série de modifications apportées aux programmes JAIN SIP Applet Phone et JAIN SIP Presence proxy afin d'intégrer l'emploi de la carte d'identité électronique dans la sécurisation de la notification de présence.

3.4.1 Algorithme développé

La notification de présence est implémentée par le traitement des requêtes REGISTER. Sécuriser la notification de présence signifie donc sécuriser l'authentification des utilisateurs à l'origine des requêtes REGISTER. Cependant, l'authentification des utilisateurs dans la version originelle des programmes concerne les requêtes de tous types et donc il a été décidé d'appliquer l'emploi de la carte eID pour authentifier les utilisateurs lors de la validation par le serveur proxy de tous les types de requêtes par souci de simplicité.

L'algorithme d'authentification qui a été développé pour employer la carte d'identité électronique dans l'authentification des utilisateurs ne se base pas sur les spécifications qui ont été étudiées précédemment. Il se base sur l'emploi des en-têtes Proxy-Authorization et Proxy-Authenticate comme la Digest Authentication. Le contenu du champ username, c'est-à-dire le nom d'utilisateur, est remplacé par le numéro de série du certificat Cert#2 de l'utilisateur et le contenu du champ response, le hashage sur le challenge, est remplacé par la signature électronique sur le challenge.

Le fonctionnement est montré à la figure 3.17. Lorsque l'user agent reçoit une réponse avec le code 407 "Proxy Authorization Required", il extrait de l'en-tête Proxy-Authenticate le challenge. Il demande à l'utilisateur sa carte d'identité électronique et son code PIN. Il extrait le certificat Cert#2 de la carte et il effectue une signature électronique avec la clef PrK#2 de celle-ci. Il crée ensuite une requête basée sur la réponse contenant un en-tête Proxy-Authorization dont le champ username contient le numéro de série du certificat extrait et dont le champ response contient la signature électronique. La requête est ensuite envoyée au serveur proxy. Celui-ci doit disposer d'un keystore contenant une copie du certificat Cert#2 de chaque utilisateur inscrit et dont l'alias est son numéro de série. Il utilise ce keystore pour retrouver le certificat Cert#2 de l'utilisateur à partir du numéro de série contenu dans le champ username de l'en-tête Proxy-Authorization. Avec ce certificat, il vérifie que la signature électronique contenue dans le champ response de ce même en-tête correspond bien avec le challenge. Si c'est le cas, l'authentification est réussie.

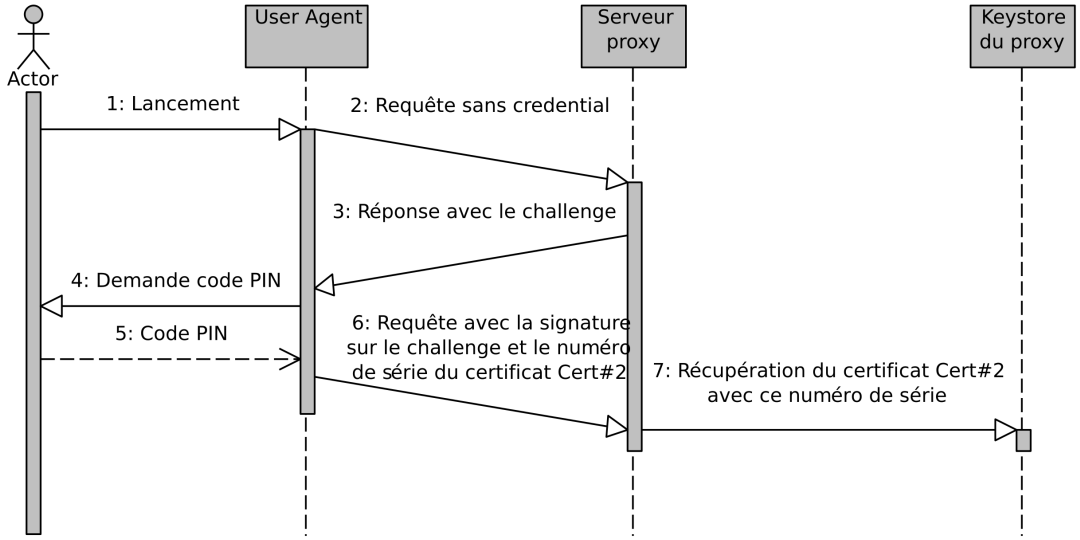

FIG. 3.17 – Fonctionnement de l'application après les modifications dues à la première réalisation

3.4.2 Modifications de JAIN SIP Applet Phone

La classe DigestClientAuthenticationMethod a été réécrite de sorte à ne plus avoir besoin du nom d'utilisateur et du mot de passe associé pour être initialisée. Lorsqu'il lui est demandé de générer le contenu du champ response de l'en-tête Proxy-Authorization de la nouvelle requête, elle crée une nouvelle fenêtre graphique qui demande à l'utilisateur d'introduire sa carte d'identité électronique dans le lecteur adéquat et d'entrer son code PIN. Ensuite, elle extrait le certificat Cert#2 de la carte et utilise cette dernière pour signer électroniquement le challenge du serveur proxy. Enfin, elle renvoie un objet contenant le numéro de série du certificat Cert#2 et la signature électronique. Les données de cet objet sont alors utilisées par le programme pour compléter l'en-tête Proxy-Authorization de la requête qui est finalement envoyée au serveur proxy.

Des modifications ont également dû être apportées à d'autres classes du programme pour gérer les modifications de DigestClientAuthenticationMethod. De plus, son interface a dû être adaptée pour être en accord avec ces modifications. Enfin le nom de la classe DigestClientAuthenticationMethod a été remplacé par EIDClientAuthenticationMethod qui est plus explicite.

3.4.3 Modifications de JAIN SIP Presence Proxy

Grâce à la possibilité de sélectionner la classe implémentant l'authentification et au caractère général des méthodes de l'interface que cette classe doit implémenter, il a été possible d'écrire une classe Java implémentant l'algorithme décrit précédemment sans avoir à modifier le reste du programme.

La nouvelle classe créée est EIDServerAuthenticationMethod. Pour être employée, elle doit être initialisée avec le chemin du keystore qui contient les certificats Cert#2 des utilisateurs inscrits avec pour alias leur numéro de série. Lorsqu'une requête lui est donnée pour vérification, elle en extrait l'en-tête Proxy-Authorization, elle utilise le numéro de série contenu dans le champ username pour retrouver le certificat dans le keystore et enfin elle vérifie que la signature électronique contenue dans le champ response est compatible avec ce certificat et le challenge. Elle renvoie la valeur booléenne True si l'authentification est réussie et False sinon.

Il a fallu également créer un petit programme pour gérer le keystore. La classe KeystoreHandler disposant d'une méthode main et se trouvant dans le paquetage gov.nist.sip.proxy.authentication a donc été créée dans ce but. Ce programme doit être lancé après avoir introduit une carte d'identité électronique dans le lecteur adéquat. Il lit le certificat Cert#2 contenu dans celle-ci et l'ajoute au keystore avec pour alias son numéro de série. Enfin, il affiche la liste de tous les alias enregistrés dans le keystore. Le fichier du keystore se nomme "keystore.jks" et se trouve dans le répertoire racine de l'application JAIN SIP Presence Proxy. S'il n'exite pas au moment du lancement du programme KeystoreHandler, un nouveau keystore de ce nom est créé.

3.4.4 Critique

Bien que l'algorithme développé lors de cette première itération implémente effectivement la sécurisation de la notification de présence à l'aide de la carte d'identité électronique, l'examen attentif de cette solution révèle trois failles de sécurité : la première concerne l'algorithme d'authentification lui-même et les deux autres concernent les programmes JAIN-SIP Applet Phone et JAIN-SIP Presence Proxy.

La faille de sécurité concernant l'algorithme est la suivante : un pirate peut intercepter les requêtes REGISTER et remplacer l'adresse IP de l'utilisateur contenue dans l'en-tête Contact par la sienne et ainsi s'enregistrer auprès du service SIP avec l'URI SIP de l'utilisateur. Cette faille dans l'algorithme rend nécessaire le développement d'un nouvel algortithme en se basant sur les spécifications qui ont été étudiées précédemment dans ce chapitre de façon à la corriger.

Les failles de sécurité concernant les programmes employés dans le cadre de ce mémoire sont héritées des programmes originels (avant les modifications liées à la première réalisation). Le détail de ces failles est le suivant :

1. Lors de la réception d'une requête avec un en-tête Proxy-Authorization, le programme JAIN SIP Presence Proxy ne vérifie pas que la valeur du champ challenge de cet en-tête correspond bien à la valeur qu'il avait générée lorsqu'il avait répondu à la requête précédente qui ne contenait pas de credential. Etant donné que c'est la valeur de ce champ qui sert à vérifier la validité de la signature électronique, un pirate peut récupérer l'en-tête Proxy-Authorization d'une requête interceptée et l'intégrer à ses requêtes REGISTER et ainsi se

faire authentifier sans difficulté.

2. Dans la version originelle, le programme JAIN SIP Presence Proxy ne mémorise pas la relation qu'il y a entre l'URI SIP et le couple nom d'utilisateur - mot de passe. Dans la première réalisation, cette faille a persisté dans l'absence de corrélation entre l'URI SIP et le numéro de série du certificat Cert#2 des utilisateurs. La conséquence est que n'importe quelle personne inscrite auprès du proxy peut être authentifiée en employant n'importe quelle URI.

3.5 Deuxième réalisation

Cette réalisation a eu pour but de corriger les failles de sécurité découvertes dans la précédente réalisation.

3.5.1 Algorithme développé

La faille de sécurité du précédent algorithme concernait la possibilité pour un pirate de modifier les en-têtes d'une requête REGISTER interceptée pour notamment usurper l'identité de l'utilisateur à l'origine de cette requête. Deux des spécifications qui ont été étudiées au début de ce chapitre peuvent appporter une solution : SAML et AIB.

SAML remplace l'en-tête Proxy-Authorization par une assertion SAML signée par une autorité SAML. Cette assertion SAML peut contenir l'adresse IP à partir de laquelle l'utilisateur doit s'authentifier. Donc si l'implémentation du serveur proxy vérifie que l'adresse IP de l'assertion SAML correspond à celle contenue dans l'en-tête Contact de la requête REGISTER, il devient difficile pour un pirate de modifier cet en-tête et d'être authentifié par la suite. Ce dernier pourrait encore rejouer une requête interceptée depuis un terminal ayant la même adresse IP, mais cela peut être contré en ajoutant aux assertions SAML une période de validité assez courte.

Malheureusement SAML exige d'ajouter au service SIP un nouveau type de serveur : l'autorité SAML, et de distribuer le certificat ou la clef publique correspondant à la clef privée que ce serveur emploie pour signer ses assertions SAML aux autres serveurs du service SIP. De plus, ce système n'exige plus que le serveur proxy envoie un challenge au user agent de l'utilisateur, donc l'authentification ne suit plus le schéma habituel (requête sans credential, réponse avec challenge, requête avec réponse avec credential) mais envoie directement une requête avec un credential (une assertion SAML) qui sera directement validée ou pas. De ce fait, il faudrait modifier considérablement le fonctionnement de JAIN SIP Presence Proxy et JAIN SIP Applet Phone.

La spécification AIB quant à elle consiste à rajouter au corps des requêtes un AIB qui contient une signature sur une sélection des en-têtes. De ce fait, il devient difficile de modifier les en-têtes de cette sélection et de s'authentifier par la suite. Comme

SAML, AIB n'a pas besoin que le serveur proxy fournisse un challenge et donc le schéma : requête sans credential, réponse avec challenge, requête avec réponse avec credential, est lui aussi inutile. Cependant, les requêtes REGISTER issues d'une même adresse IP sont toutes semblables ou peu s'en faut, il est donc possible de rejouer une de ces requêtes depuis un terminal disposant de la même adresse IP pour s'enregistrer avec l'URI d'une autre personne. La solution serait d'intégrer dans la sélection des en-têtes à signer un élément qui soit unique pour chaque requête afin d'empêcher le rejeu, élément qui doit faire l'objet d'une vérification lors du traitement de la requête par le serveur proxy. L'élément qui a été choisi est un challenge fourni par le serveur proxy afin de ne pas avoir à modifier le schéma de l'authentification et de disposer d'un élément aléatoire dans chaque requête qui empêche qu'elle soit rejouée.

L'algorithme qui a été développé s'est donc basé sur la spécification AIB. Il continue à se servir des en-têtes Proxy-Authorization et Proxy-Authenticate. Lorsqu'un utilisateur souhaite envoyer une requête (REGISTER ou autre) au serveur proxy, il envoie une première instance de cette requête dépourvue de credential et le serveur proxy lui envoie une réponse contenant un challenge. Une fois cette réponse reçue, l'user agent de l'utilisateur va créer une nouvelle instance de la requête avec un en-tête Proxy-Authorization qui ne contient plus de champ username ni de champ response devenus tous deux obsolètes. Il crée un AIB selon le schéma de la figure 3.18. Il sélectionne ensuite les en-têtes à signer (il est recommandé que Proxy-Authorization et Contact en fassent partie), il génère la signature électronique sur cette sélection avec la carte d'identité électronique de l'utilisateur et enfin il crée l'AIB avec la sélection des en-têtes, la signature et le certificat Cert#2. Il lui reste à insérer l'AIB dans le corps de la requête et à l'envoyer au serveur proxy.

La vérification du serveur proxy est représentée à la figure 3.19. Lors de la réception de la requête avec l'AIB et l'en-tête Proxy-Authorization, le serveur proxy commence par vérifier que les en-têtes de la sélection contenue dans le AIB sont équivalents avec ceux qui sont dans la requête elle-même. Il récupère ensuite dans le keystore le certificat Cert#2 de l'utilisateur dont l'alias correspondant n'est plus le numéro de série mais l'URI SIP de l'utilisateur ce qui permet de corriger la faille de sécurité due à l'absence de relation entre les certificats et les URI. Il lui reste à vérifier avec ce certificat que la signature de l'AIB est correcte. Si toutes les vérifications sont un succès, l'authentification est réussie.

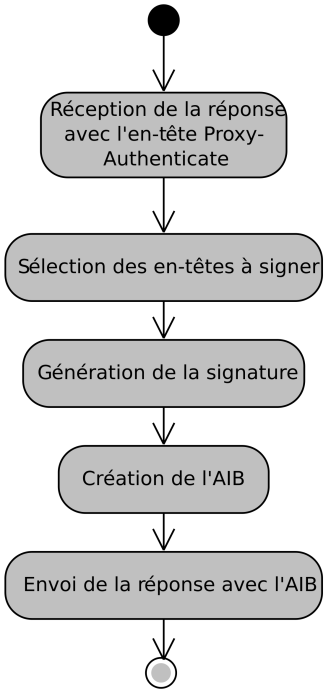

FIG. 3.18 – Diagramme d'activité représentant la génération d'un AIB

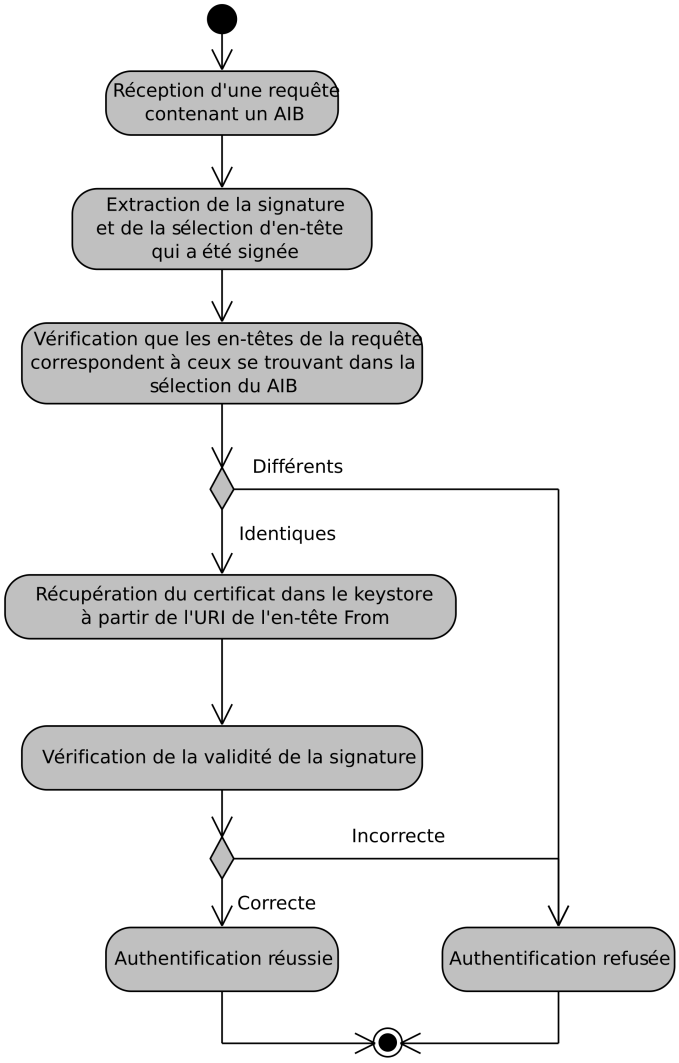

FIG. 3.19 – Diagramme d'activité représentant la vérification d'un AIB

3.5.2 Modification de JAIN SIP Applet Phone

L'emploi de la classe EIDClientAuthenticationMethod a été abandonné au profit d'une classe AIBGenerator créée dans le paquetage gov.nist.applet.phone.ua.authentication. Cette nouvelle classe n'implémente pas l'interface ClientAuthenticationMethod. Une nouvelle instance de cette classe est créée à chaque fois qu'une réponse 407 "Proxy Authorization Required". Elle contient une seule méthode : generate, qui effectue les opérations suivantes :

1. Ajout d'un en-tête Proxy-Authorization à la requête,
2. sélection des en-têtes à signer,
3. création d'une fenêtre pour demander à l'utilisateur d'introduire sa carte d'identité dans le lecteur adéquat et de composer son code PIN,
4. extraction du certificat Cert#2,
5. génération de la signature avec la carte eID, le code PIN et la sélection d'en-têtes,
6. création de l'AIB avec la sélection d'en-tête, le certificat Cert#2 et la signature électronique,
7. insertion de l'AIB dans la requête.

Après l'appel de cette méthode, il ne reste plus au programme qu'à envoyer la requête au serveur proxy.

3.5.3 Modification de JAIN SIP Presence Proxy

La classe gov.nist.sip.proxy.authentication a été modifiée de façon à ce que lorsqu'il lui est demandé d'authentifier l'utilisateur à l'origine d'une requête elle extrait l'AIB de la requête, vérifie que les en-têtes de la sélection sont équivalents à ceux de la requête, récupère le certificat Cert#2 dans le keystore grâce à l'URI SIP de l'en-tête From de la requête et se sert de ce dernier pour vérifier la signature de l'AIB. Si toutes les vérifications sont un succès, l'authentification est réussie.

Le programme KeystoreHandler a lui aussi été modifié de sorte que les alias des certificats ne soient plus leur numéro de série mais l'URI SIP de leur propriétaire. De cette façon, une des failles de sécurité de la première réalisation a été corrigée.

Enfin, la dernière faille de sécurité de ce programme, celle concernant l'absence de vérification sur la valeur du challenge contenu dans l'en-tête Proxy-Authorization, a été corrigée. Pour ce faire un buffer destiné à recevoir les challenges générés par le programme a été ajouté à ce dernier. Chaque fois qu'une requête avec un en-tête Proxy-Authorization est reçu, le programme vérifie que la valeur du challenge de cet en-tête est présente dans ce buffer, si ce n'est pas le cas l'authentification de l'utilisateur échoue. Afin d'éviter un débordement de la mémoire, le buffer a une taille maximum (50 dans le cadre de ce mémoire) et le fait de lui rajouter un challenge

lorsqu'il est plein entraîne l'écrasement du plus vieil enregistrement. De plus, chaque fois que le programme vérifie avec succès la présence d'un challenge dans le buffer, ce dernier désormais inutile est effacé. Grâce à ce buffer, il est désormais difficile de rejouer une requête interceptée.

3.5.4 Critique

L'algorithme d'authentification utilisé dans cette seconde réalisation permet à la fois d'authentifier l'utilisateur et d'empêcher la modification et la réutilisation de requêtes par un pirate. De plus, les failles de sécurité présentes dans les programmes originels ont été corrigées.

Cet algorithme d'authentification et son implémentation dans les programmes JAIN SIP Applet Phone et JAIN SIP Presence Proxy peut être considéré comme rendant impossible l'usurpation d'identité dans le cadre de la notification de présence SIP à condition que soient supposées vraies les conditions suivantes :

- Le cryptage RSA est incassable,
- la carte d'identité de l'utilisateur n'a pas été volée et son code PIN reste secret,
- les cartes d'identité électroniques sont infalsifiables,
- la personne en charge de l'enregistrement des certificats Cert#2 dans le keystore est fiable.

3.6 Tests de validation

Afin de s'assurer que l'implémentation de l'algorithme d'authentification des utilisateurs avec la carte d'identité électronique est correcte, des tests ont été effectués sur un ordinateur de type PC fontionnant sous Windows XP et équipé d'un lecteur de carte d'identité électronique Zetes ACR38U. Les tests réalisés sont les suivants :

D'abord des tests qui devaient réussir et qui ont consisté à s'authentifier avec différentes cartes d'identité électroniques dont le certificat Cert#2 avait été enregistré dans le keystore. L'authentification se faisait bien sûr avec l'URI SIP qui sert d'alias au certificat Cert#2 de la carte eID utilisée dont une copie est stockée donné dans le keystore. Tous ces tests ont été un succès.

Ensuite, des tests qui devaient échouer parce qu'ils mettaient en scène des cas où l'authentification ne devait pas réussir. Ces test étaient :

- S'authentifier avec une URI inexistante et une carte eID enregistrée,
- s'authentifier avec une URI inexistante et une carte eID non enregistrée,
- s'authentifier avec une URI existante et une carte eID non enregistrée,
- s'authentifier avec une URI existante et une carte eID enregistrée mais pas celle qui correspond à l'URI.

Tous ces tests ont échoués.

Chapitre 4
Réalisations futures possibles

Le but de ce chapitre est de présenter des réalisations qui pourraient faire suite à ce mémoire.

4.1 Créer un Webservice pour l'inscription des utilisateurs

L'implémentation de l'algorithme d'authentification des utilisateurs avec la carte d'identité électronique développé lors de ce travail afin d'utiliser la carte eID pour sécuriser la notification de présence requiert l'usage d'un keystore. Ce dernier sert à conserver une copie des certificats Cert#2 des utilisateurs. Il faut donc que lorsqu'un nouvel utilisateur s'inscrit auprès du service SIP, son certificat Cert#2 soit enregistré dans le keystore.

L'application qui a été développée pour effectuer l'inscription des nouveaux utilisateurs exige que la carte eID d'un nouvel utilisateur soit insérée dans le lecteur de carte du terminal hébergeant le keystore. Cette solution est acceptable pour un service SIP de petite taille où les utilisateurs sont géographiquement proches des serveurs comme par exemple dans le cas d'une petite entreprise où les employés communiqueraient entre eux avec un service SIP. Par contre, ce n'est plus acceptable pour un service de grande taille où les utilisateurs sont éloignés des serveurs.

Une solution plus efficace, serait de créer un Webservice qui permettrait aux utilisateurs de s'inscrire auprès du service SIP. Au cours de cette inscription, le nouvel utilisateur devrait envoyer au Webservice son certificat Cert#2, l'URI qu'il souhaite utiliser et une signature effectuée avec sa carte eID. Le Webservice vérifierait la validité du certificat Cert#2 avec le protocole OCSP puis la validité de la signature électronique pour s'assurer que c'est bien le propriétaire de la carte qui l'utilise pour s'inscrire et enfin que l'URI désirée est disponible. Si toutes ces vérifications réussissaient, le Webservice pourrait procéder à l'enregistrement du certificat Cert#2 dans le keystore avec pour alias l'URI demandée.

En employant un Webservice de ce type, il serait possible d'établir un service SIP employant la carte d'identité électronique pour sécuriser la notification de présence déployé sur une grande échelle.

4.2 Employer SIPS

Le protocole SIP présente l'inconvénient que ses requêtes passent en clair. Il est certes possible de crypter le corps des messages notamment avec S/MIME mais les en-têtes, eux, ne peuvent pas être cryptés. Il est donc possible d'intercepter les requêtes, notamment les requêtes REGISTER, et de les réutiliser après avoir modifié certains de leurs en-têtes pour effectuer des attaques de sécurité comme l'usurpation d'identité. La solution qui a été employée dans le cadre de ce mémoire a été de signer une sélection des en-têtes afin que toute manipulation sur leur contenu entraîne un échec de l'authentification.

Une solution plus efficace serait de crypter l'intégralité des requêtes SIP (le corps et les en-têtes) afin qu'en cas d'interception des requêtes, celles-ci soient incompréhensibles et donc non réutilisables. Cela est possible avec la version sécurisée de SIP : SIPS. Cette variante du protocole spécifie que les communications entre les entités du système doivent être sécurisée avec le protocole TLS.

Le protocole TLS est une variante du protocole SSL et est défini dans la RFC 4346 [3]. Il permet à deux entités réseaux de générer une clef symétrique pour crypter leurs communications et optionnellement de s'authentifier entre elles à l'aide de clefs assymétriques privées et de certificats. Il est donc possible que l'utilisateur s'authentifie avec sa carte eID dans le cadre de ce protocole.

L'emploi du protocole SIPS permettrait d'abandonner l'usage des en-têtes Proxy-Authenticate et Proxy-Authorization ainsi que des AIB puisque l'utilisateur serait authentifié dès l'établissement de la connexion entre lui et le proxy et donc avant que sa première requête n'arrive à ce dernier. De plus, les communications ne seraient pas seulement cryptées entre l'utilisateur et le proxy mais entre lui et chacun de ses correspondants avec la possibilité d'effectuer des authentifications entre eux. En bref, l'usage du protocole SIPS permet d'accroître de façon significative la sécurité du service SIP.

4.3 Authentifier les correspondants

Le protocole SIP a pour ambition de devenir le principal support des communications électroniques dans un proche avenir et donc d'éclipser les communications téléphoniques. Les communications téléphoniques sont basées sur l'emploi de numéros. Ces numéros sont attribués par des compagnies téléphoniques qui peuvent être considérées comme fiables et qui garantissent la corrélation entre le numéro et l'individu ou la société à qui il a été attribué[5]. Les communications SIP ou sessions SIP sont basées sur l'emploi d'une URI (ex : sgamby :fundp.ac.be). Ces URI sont attribuées par les administrateurs du service SIP du domaine réseau de l'utilisateur (ex : fundp.ac.be), ces derniers ne peuvent pas être considérés d'office comme fiables vu que certains d'entre eux ne prennent pas la peine de vérifier l'identité des personnes qui demandent une URI. Le fait d'employer une session SIP plutôt qu'une communication téléphonique entraîne donc une incertitude sur l'identité du propriétaire de l'URI SIP et à plus forte raison sur l'identité de la personne qui utilise cette URI au cours de la sesssion.

Dans le cadre de ce mémoire, une authentification entre les serveurs proxy et les utilisateurs a été développée. Cette authentification requiert le stockage dans un keystore présent chez le serveur proxy des certificats Cert#2 des utilisateurs, ces certificats ayant pour alias l'URI SIP de leur propriétaire respectif. Il serait donc possible d'installer un service d'annuaire basé sur ce keystore qui permettrait de fournir le certificat Cert#2 correspondant à une URI SIP et donc de garantir l'identité du propriétaire de cette URI. De plus, l'authentification eID garantit que seul le propriétaire d'une URI SIP peut enregistrer sa présence avec celle-ci, assurant du même coup l'identité de l'utilisateur de l'URI au cours d'une session SIP. Il est donc possible à une personne d'obtenir des garanties sur l'identité de son correspondant à condition que celui-ci utilise le service SIP d'un domaine réseau employant l'authentification eID développée au cours de ce travail et disposant d'un service d'annuaire tel que décrit plus haut. Si ce n'est pas le cas, cette personne ne peut pas bénéficier de ce genre de garanties.

Il existe une solution qui permettrait à une personne d'obtenir des garanties sur l'identité d'un de ses correspondants dont le service SIP n'emploie pas l'authentification eID décrite dans ce travail : l'authentification eID entre utilisateurs. La RFC de SIP [11] spécifie qu'un utilisateur peut s'authentifier auprès d'un autre en employant la Digest Authentication comme s'il s'authentifiait auprès d'un serveur proxy. La différence est que les noms des en-têtes Proxy-Authorization et Proxy-Authenticate sont remplacés respectivement par Authorization et WWW-Authenticate et que l'authentification peut se faire dans les deux sens.

Intégrer la carte eID dans l'algorithme d'authentification entre utilisateurs est légèrement plus complexe que pour celui entre utilisateur et serveur proxy parce qu'il n'y a pas de connaissance pré-supposée entre les entités et donc l'utilisateur qui demande à son correspondant de s'identifier ne possède pas nécessairement une

[5]Exception faite bien sûr des cartes de GSM prépayées

copie du certificat Cert#2 de ce dernier comme c'est le cas pour le serveur proxy. Dès lors, un utilisateur qui souhaite s'authentifier en répondant au challenge de son correspondant doit lui renvoyer son certificat Cert#2 en plus de la signature sur le challenge. L'emploi de l'algorithme AIB [10] est une solution puisque le SignedData contenu dans le corps de la requête contient la signature et le certificat qui lui correspond. Un utilisateur exigeant de son correspondant qu'il s'authentifie, recevra de celui-ci une requête contenant un AIB. Il lui suffira d'en extraire la signature et le certificat, de vérifier que la signature correspond bien au certificat et enfin de consulter les informations d'identité contenues dans celui-ci pour être certain de l'identité de son correspondant. Enfin, il sera également possible de vérifier la validité du certificat en faisant appel au service de publication de CRL ou au serveur OCSP.

Le sequence diagram de la figure 4.20 présente l'échange des messages entre les entités du système lors de l'envoi d'une requête INVITE d'un utilisateur A à un utilisateur B. Pour commencer, l'utilisateur A s'authentifie auprès de son serveur proxy avec l'algorithme AIB et sa carte eID (il envoie une première requête sans credential, reçoit une réponse avec un challenge et envoie une requête avec la signature eID sur le challenge). Ensuite, sa requête contenant une demande d'authentification du correspondant avec challenge est relayée par les serveurs proxy jusqu'à l'utilisateur B qui renvoie par le chemin inverse une réponse contenant sa signature eID sur le challenge et son propre challenge d'authentification. L'utilisateur B n'a pas à s'authentifier auprès de son serveur proxy parce qu'il renvoie une réponse. L'utilisateur A vérifie la signature de B, signe le challenge de celui-ci et envoie directement une requête à B avec sa signature. Pour finir, l'utilisateur B vérifie la signature de A et la session SIP peut se poursuivre avec chacun des utilisateurs ayant authentifié l'autre.

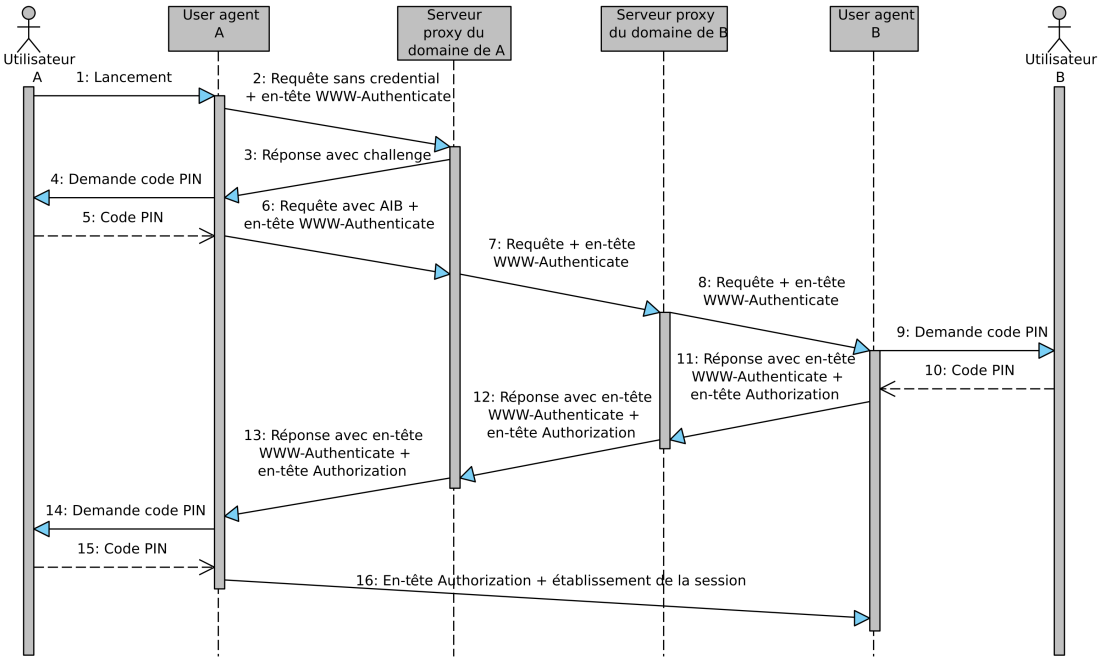

FIG. 4.20 – Sequence diagram présentant l'échange des messages lors de l'envoi d'une requête par un utilisateur A à un utilisateur B avec authentification des utilisateurs.

4.4 Employer SAML à la place de AIB

Lors du développement de l'algorithme d'authentification, il a été décidé de s'inspirer de la spécification AIB plutôt que de la spécification SAML parce que cette dernière imposait notamment l'ajout d'un nouveau type de serveur au système : l'autorité SAML.

Le service SIP employé dans le cadre de ce mémoire était de taille modeste et employait un seul serveur qui assurait les services proxy, registrar et location. Il était donc plus simple d'améliorer la fonction d'authentification des utilisateurs que de créer un nouveau serveur pour assurer seul cette fonction.

Cependant, cela change avec un service SIP de grande taille disposant de nombreux serveurs proxy et autres. Tout d'abord, l'emploi d'une autorité SAML permettrait de réduire la charge des serveurs proxy. En effet, l'enregistrement de la notification de présence d'un utilisateur avec la sécurisation développée dans ce travail nécessite deux requêtes successives de l'utilisateur : une première sans credential à laquelle le serveur répond en fournissant un challenge d'authentification et une deuxième contenant un credential qui est la réponse au challenge du serveur. Le fait d'employer une autorité SAML, permettrait à l'utilisateur de notifier sa présence avec une seule requête qui contiendrait directement un credential acceptable pour les serveurs proxy : une assertion SAML. Cela permettrait donc de réduire la charge sur les serveurs proxy.

Ensuite, le fait de migrer la fonction d'authentification des utilisateurs dans des serveurs différents permettrait d'améliorer cette fonction sans avoir à modifier les serveurs proxy. Donc si les administrateurs du service SIP décidaient par exemple de modifier l'algorithme d'authentification ou de permettre l'emploi de différents algorithme, il suffirait de modifier les autorités SAML qui sont moins nombreuses que les serveurs proxy et qui ont une charge moindre et sont donc plus simple à modifier.

Enfin, cela permettrait de mieux se prévenir des attaques des pirates. Le point faible des authentifications avec la carte d'identité électronique est le keystore contenant une copie des certificats Cert#2 des utilisateurs. Parvenir à insérer un faux certificat Cert#2 dans ce keystore permettrait à un pirate d'employer le service et d'abuser les autres utilisateurs. Les serveurs s'occupant de l'authentification sont donc susceptibles de subir des attaques (virus, vers, etc.) afin de forcer l'accès à leur keystore. Si l'authentification est assurée directement par les serveurs proxy, ces derniers risquent de voir leur qualité de service décliner et même disparaître sous l'effet des attaques des pirates. De plus, leur adjoindre des protections efficaces contre ces attaques (pare-feu, antivirus) risque de diminuer leur capacité de traitement et donc de diminuer leur qualité de service. Par contre si ce sont des autorités SAML qui assurent l'authentification des utilisateurs, étant donné que leur charge de travail est moins importante que celle des serveurs proxy, il est possible de leur adjoindre des protections gourmandes en ressources matérielles sans que les utilisateurs soient lésés. De plus, en cas de panne, les utilisateurs déjà enregistrés auprès du service SIP ne seront pas lésés.

4.5 Etendre l'usage de la carte eID

Le protocole SIP est conçu sur le même modèle que le protocole HTTP, c'est à dire des requêtes composées d'en-têtes et d'un corps séparés par une ligne vide. L'algorithme d'authentification qui a été développé dans le cadre de ce mémoire, l'ajout d'un AIB contenant une signature effectuée sur une sélection des en-têtes des requêtes avec la carte d'identité électronique, pourrait être utilisé pour sécuriser l'authentification de l'utilisateur non seulement dans le protocole HTTP lui-même mais dans tous les protocoles réseaux qui sont basés sur son modèle.

4.6 Exploiter l'information de présence sécurisée par eID

L'algorithme d'authentification développé lors de ce travail permet de sécuriser la notification de présence dans des protocoles réseaux (SIP et autres) avec la carte d'identité électronique. De ce fait, l'information de présence d'un utilisateur dans un service SIP donné et durant une période donnée dispose désormais d'un certain

crédit et pourrait donc être exploitée.

Par exemple, il serait possible d'exploiter juridiquement cette information dans le cadre de délits informatiques. Si des personnes se connectent à un service Internet (ex : un forum) avec leur carte eID, il devrait être possible de les poursuivre s'il commettait un délit pendant leur utilisation de ce service (propos raciste, échange de matériel pornographique pédophile, complot terroriste, etc.).

Il serait donc intéressant que de futurs travaux s'intéressent aux différentes façons dont l'information de présence sécurisée par la carte d'identité électronique pourrait être exploitée notamment dans un cadre juridique.

Conclusion

Dans le cadre de ce travail, une solution pour sécuriser la notification de présence du protocole SIP a été développée. Cette solution consiste à intégrer dans les requêtes REGISTER, les vecteurs de la notification de présence sous SIP, un Authenticated Identity Body (AIB). Ce dernier permet à l'utilisateur de s'authentifier à l'aide d'une signature électronique réalisée avec sa carte eID. De plus, il protège les en-têtes des requêtes auxquelles il a été ajouté contre les modifications qu'un pirate pourrait réaliser afin d'usurper l'identité de l'utilisateur.

Les concepts qui ont été examinés au cours de ce mémoire peuvent être réutilisés pour étendre l'utilisation de la carte d'identité électronique à d'autres domaines que la seule sécurisation de la notification de présence. Il est ainsi possible d'employer la carte d'identité électronique pour authentifier l'utilisateur auprès du serveur proxy lors de l'envoi de toutes les requêtes et pas seulement les requêtes REGISTER ; d'utiliser la carte eID pour permettre aux utilisateurs d'une session SIP de s'authentifier entre eux ; et enfin la carte d'identité électronique peut être employer pour identifier son propriétaire dans le cadre d'autres protocoles réseaux.

En définitive, ce travail a atteint les objectifs qui lui avaient été fixés et il ouvre la voie à l'intégration de la carte eID dans la notification de précence de nombreux autres protocoles réseaux.

Bibliographie

[1] Mark Bartel, John Boyer, Barb Fox, Brian LaMacchia, and Ed Simon. *XML-Signature Syntax and Processing*. W3C, `http://www.w3.org/TR`, 2002.

[2] Conor P. Cahill, John Hugues, Hal Lockhart, Michael Beach, Rebekah Metz, Rick Randall, Thomas Wisniewski, Irving Reid, Paula Austel, Maryann Hondo, Michael McIntosh, Tony Nadalin, Nick Ragouzis, Scott Cantor, Peter C Davis RL 'Bob' Morgan, Jeff Hodges, Frederick Hirsch, John Kemp, Paul Madsen, Steve Anderson, Prateek Mishra, John Linn, Rob Philpott, Janah Moreh, and Anne Anderson. *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) v2.0*. OASIS, `http://www.oasis-open.org`, March 2004.

[3] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. Request for Comments 4346, Network Working Group, `http://www.ietf.org`, April 2006.

[4] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart. HTTP Authentication : Basic and Digest Access Authentication. Request for Comments 2617, Network Working Group, `http://www.ietf.org`, June 1999.

[5] C. Jennings, J. Peterson, and M. Watson. Private Extension to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks. Request for Comments 3325, Network Working Group, `http://www.ietf.org`, November 2002.

[6] RSA Laboratories. PKCS #7. Public-Key Cryptography Standards (PKCS), RSA Data Security, `http://www.rsasecurity.com/rsalabs`, November 1993.

[7] RSA Laboratories. PKCS #15 v1.1 : Cryptographic Token Information Syntax Standard. Public-Key Cryptography Standards (PKCS), RSA Data Security, `http://www.rsasecurity.com/rsalabs`, June 2000.

[8] Frédéric Leprieur and Marc Stern. *General Technical Specification Belpic Applet v1.4a*. SchlumbergerSema Systèmes SA, 2002.

[9] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. Request for Comments 2560, Network Working Group, `http://www.ietf.org`, June 1999.

[10] J. Peterson. Session Initiation Protocol (SIP) Authenticated Identity Body (AIB) Format. Request for Comments 3893, Network Working Group, `http://www.ietf.org`, September 2004.

[11] J. Rosenberg, H. Schulzrinne, Columbia U., G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP : Session Initiation Protocol. Request for Comments 3261, Network Working Group, `http://www.ietf.org`, June 2002.

[12] Marc Stern. *Belgian Electronic Identity Card Content*. Zetes, December 2003.

[13] T. Ylonen and C. Lonvik. The Secure Shell (SSH) Authentication Protocol. Request for Comments 4252, Network Working Group, `http://www.ietf.org`, January 2006.

[14] T. Ylonen and C. Lonvik. The Secure Shell (SSH) Connection Protocol. Request for Comments 4254, Network Working Group, `http://www.ietf.org`, January 2006.

[15] T. Ylonen and C. Lonvik. The Secure Shell (SSH) Protocol Architecture. Request for Comments 4251, Network Working Group, `http://www.ietf.org`, January 2006.

[16] T. Ylonen and C. Lonvik. The Secure Shell (SSH) Transport Layer Protocol. Request for Comments 4253, Network Working Group, `http://www.ietf.org`, January 2006.

Annexes

Annexe A
Première version du galop d'essai

Le but de cette annexe est de présenter la première version du galop d'essai de ce mémoire et d'expliquer pourquoi elle a été abandonnée au profit de la version présentée au chapitre 1 (page 14).

Documentation

Le galop d'essai avait pour but de se familiariser avec la carte d'identité électronique. La première version visait à intégrer l'usage de la carte eID dans un autre protocole que SIP : le protocole SSH. L'idée étant que l'implémentation d'une authentification de l'utilisateur basée sur l'emploi de la carte d'identité électronique serait plus simple avec le protocole SSH qu'avec le protocole SIP.

La première étape a été de se documenter sur le protocole SSH afin de comprendre son fonctionnement et de déterminer le moyen qui serait employé pour introduire l'authentification eID dans son fonctionnement. Pour ce faire, la RFC décrivant la structure du protocole SSH (RFC 4251 [15]) a été étudiée. Celle-ci explique que le protocole SSH permet à un utilisateur de se connecter à distance à un terminal et de s'en servir comme s'il était connecté localement (lancement de programme, édition des fichiers, etc.). Ce protocole emploie une architecture client-serveur où le serveur est un programme démon qui est chargé dans le terminal sur lequel l'utilisateur garde ses fichiers ; le client, quant à lui, est un programme chargé dans le terminal sur lequel l'utilisateur est loggué et qui permet de contacter le serveur.

Le protocole SSH est composée de trois couches successives, chacune possède son utilité propre et est décrite dans une RFC différente.

La couche Transport : elle fournit l'authentification du serveur ainsi que la confidentialité et l'intégrité des messages échangés entre le client et le serveur. Elle est décrite dans la RFC 4253 [16].

La couche User Authentication : elle permet à l'utilisateur de s'authentifier auprès du serveur. Elle est décrite dans la RFC 4252 [13].

La couche Connection : elle gère le tunnel crypté qui a été établi entre le client et le serveur. Elle est décrite dans la RFC 4254 [14].

La couche User Authentication est la plus intéressante dans le cadre de ce mémoire puisqu'elle s'occupe de permettre l'authentification de l'utilisateur. Cette authentification peut être de quatre types différents :

1. None : aucune authentification n'est effectuée ;

2. publickey[6] : l'utilisateur s'authentifie en fournissant une signature électronique, son nom d'utilisateur et sa clef publique (ou un certificat la contenant) ;
3. password : l'utilisateur s'authentifie en fournissant son nom d'utilisateur et son mot de passe ;
4. hostbased : idem que publickey sauf que l'utilisateur signe en employant la clef privée du terminal qu'il utilise.

La méthode d'authentification publickey permettrait d'employer la carte eID. L'utilisateur emploierait sa carte pour effectuer la signature et enverrait son certificat Cert#2 ou la clef publique de ce dernier. Il est également important de signaler que la RFC recommande que le serveur possède une base de donnée contenant les noms d'utilisateurs valides et les clefs publiques (ou les certificats) qui y sont associés afin de vérifier la validité des informations d'authentification.

Réalisation

Une fois que la possibilité d'employer la carte eID dans le protocole SSH a été avérée d'un point de vue théorique, il a été possible de passer à la suite : implémenter l'emploi de la carte eID dans une application employant SSH.

Il a donc fallu trouver une API Java implémentant SSH et dont le code source soit facilement disponible. Une seule a été trouvée : J2SSH qui peut être téléchargée sur le site `http://sourceforge.net`. Une fois le code source de l'application téléchargé, le travail a pu commencer mais il a été rapidement ralenti par un grave problème : le manque de documentation.

L'application J2SSH est assez complexe. Il serait possible de s'y retrouver facilement si la documentation était bien faite, ce qui est le cas pour les classes implémentant le côté client de l'application. La figure 5.21 montre d'ailleurs la javadoc de la classe SshClient qui est assez bien documentée. Par contre, les classes du coté serveur, elles, sont peu ou pas documentées. La figure 5.22 montre la javadoc de la classe SshDaemon qui ne contient que des mentions “Document me”.

[6] En un seul mot dans la RFC

FIG. 5.21 – Capture d'écran montrant la javadoc de la classe SshClient de l'API J2SSH

FIG. 5.22 – Capture d'écran montrant la javadoc de la classe SshDaemon de l'API J2SSH

Une autre tentative pour obtenir une documentation suffisante a été effectuée en employant le programme Doxygen, mais comme le code du coté serveur est presque exempt de commentaire, la documentation ainsi générée ne valait pas mieux que la javadoc.

Etant donné la complexité du code et le manque de documentation valable, il a fallu se rendre à l'évidence qu'intégrer la carte eID dans l'API J2SSH exigerait plus de temps et de travail qu'il ne pouvait en être accordé à un galop d'essai. Il a été également envisagé de créer une application à partir des RFC de SSH, mais là encore le temps aurait fait défaut. Finalement, il a été décidé d'abandonner l'idée d'intégrer l'emploi de la carte eID dans le protocole et il a été opté pour un galop d'essai.

Cependant, l'intégration de la carte eID dans le protocole SSH demeure possible. Elle n'a été abandonnée uniquement parce que ce n'était pas là l'objet principal de ce mémoire et que le temps aurait manqué. Il serait intéressant que de futurs travaux soient dédiés à l'emploi d'une authentification eID dans le protocole SSH.

Annexe B
Fonctionnement de JAIN SIP Presence Proxy

Le but de cette annexe est de présenter sommairement le fonctionnement du programme JAIN SIP Presence Proxy avec les modifications qui lui ont été apportées au cours de ce mémoire.

L'application doit être lancée en lui fournissant le chemin du fichier de configuration XML précédé de "-CF". Une fois l'application lancée, la fenêtre présentée à la figure 6.23 apparaît. Aucun service n'est encore assuré par l'application à ce moment là.

FIG. 6.23 – La fenêtre de JAIN SIP Presence Proxy telle qu'elle apparaît au démarrage

En cliquant sur le bouton "Menu" en haut à gauche, un onglet intitulé "Configuration" apparaît à son tour et en cliquant sur celui-ci une nouvelle fenêtre présentée à la figure 6.24 apparaît. Cette nouvelle fenêtre propose un menu à onglets qui permet de configurer l'application en modifiant le fichier XML de configuration. L'onglet qui apparaît en premier se nomme "SIP Stack" et permet de choisir notamment l'adresse réseau du terminal. Dans cet exemple, l'adresse choisie est "localhost". Deux autres onglets sont intéressants : l'onglet "Registrar" et l'onglet "Authentication". L'onglet "Registrar" est montré à la figure 6.25, il permet notamment de choisir si l'application va implémenter le service registrar ou pas et les domaines réseaux qu'elle va couvrir. L'onglet "Authentication" quant à lui est montré à la figure 6.26, il permet d'activer ou non l'authentification de l'utilisateur lors de la validation d'une requête, de choisir la classe Java qui prendra en charge l'authentification et d'entrer le chemin

de l'authentication file. Le fait de ne pas sélectionner le schéma d'authentification empêche le bon fonctionnement de l'authentification. Enfin, le bouton "Apply" qui apparaît au bas de chaque onglet du menu de configuration permet d'appliquer les changements et de revenir à la fenêtre principale.

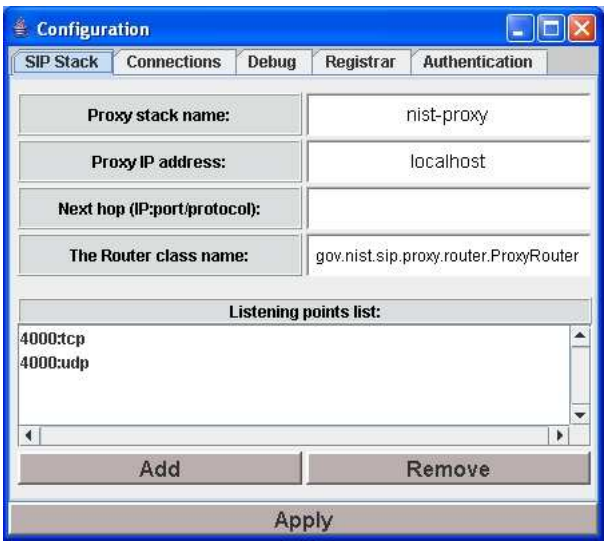

FIG. 6.24 – L'onglet "SIP Stack" du menu de configuration de cette application

Une fois l'application configurée et les changements appliqués, le service proxy (et le service registrar s'il a été activé) peut être lancé en cliquant sur le bouton "Start the proxy" de la fenêtre principale (voir figure 6.23). Une fois, le service proxy lancé, le bouton "Start the proxy" est mis en surbrillance et est renommé "Stop the proxy" comme montré à la figure 6.27. Le fait de cliquer à nouveau sur ce bouton désactive le service proxy (et le service registrar si activé) et la fenêtre redevient celle présentée à la figure 6.23. De plus, si le service registrar est activé, l'URI SIP de tous les utilisateurs ayant notifié leur présence apparaît dans le menu déroulant. L'application peut être arrêtée en cliquant sur le bouton "Quit" en haut à droite de la fenêtre principale (voir les figures 6.23 et 6.27).

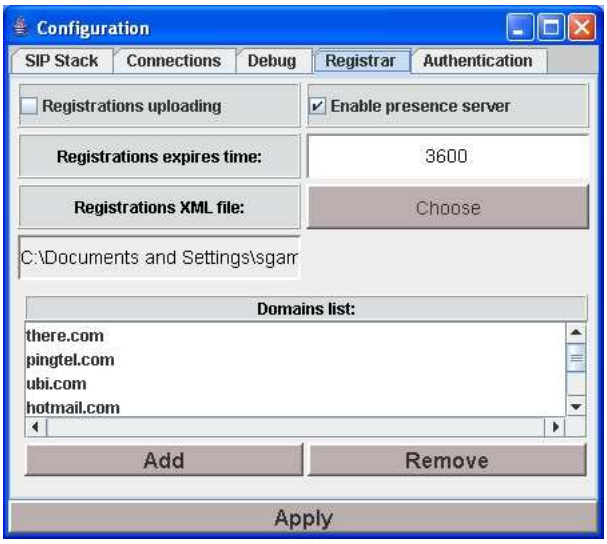

FIG. 6.25 – L'onglet "Registrar" du menu de configuration de cette application

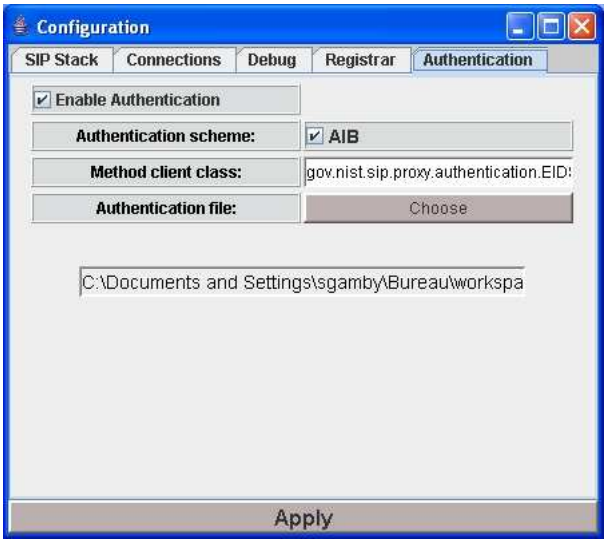

FIG. 6.26 – L'onglet "Authentication" du menu de configuration de cette application

FIG. 6.27 – La fenêtre de JAIN SIP Presence Proxy telle qu'elle est après le lancement du service proxy

Annexe C
Fonctionnement du programme JAIN SIP Applet Phone

Le but de cette annexe est de présenter sommairement le fonctionnement du programme JAIN SIP Applet Phone avec les modifications qui lui ont été apportées au cours de ce travail.

La figure 7.28 montre la fenêtre qui apparaît au lancement de l'application après avoir cliqué sur le bouton "Menu" qui se trouve en haut à gauche. Le menu qui apparaît présente plusieurs options dont les plus intéressantes sont "Configuration" et "Register".

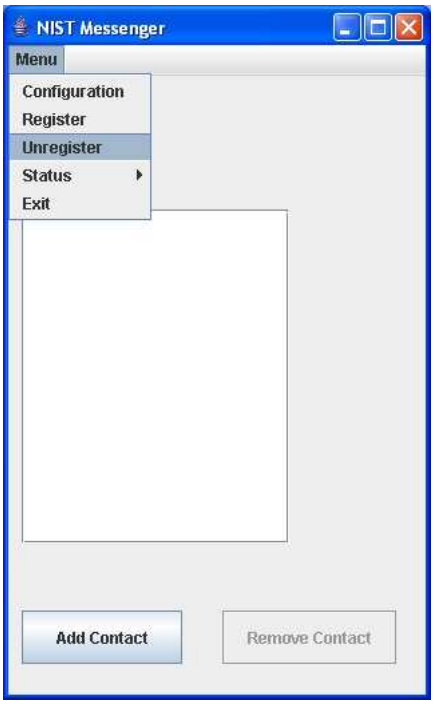

FIG. 7.28 – La fenêtre de l'application JAIN SIP Applet Phone avec le menu déroulé

Cliquer sur l'option "Configuration" fait apparaître une nouvelle fenêtre qui permet de configurer l'application. Cette fenêtre est montrée à la figure 7.29. Elle permet de choisir l'URI avec laquelle l'utilisateur va s'enregistrer, l'adresse réseau du serveur proxy à contacter, le port de ce serveur et de décider si la communication audio se fera en temps réel ou par messages vocaux. Une fois la configuration mise à jour, cliquer sur le bouton "Apply changes" permet d'appliquer les changements et de revenir à la fenêtre principale. Contrairement à l'application JAIN SIP Presence Proxy, JAIN SIP Applet Phone n'utilise pas de fichier de configuration XML ou

autre. De ce fait, la configuration est perdue à chaque arrêt de l'application et doit donc être refaite à chaque lancement de l'application.

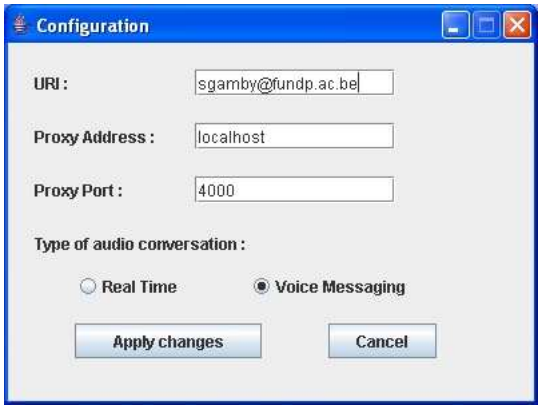

FIG. 7.29 – Le menu de configuration de cette application

Cliquer sur l'option "Register" provoque l'enregistrement de l'utilisateur auprès du serveur proxy. Si le serveur proxy exige une authentification de l'utilisateur, une nouvelle fenêtre montrée à la figure 7.30 apparaît. Cette fenêtre demande à l'utilisateur d'insérer sa carte d'identité électronique dans le lecteur adéquat et de composer son code PIN. Une fois cela fait, l'utilisateur clique sur le bouton "OK" pour lancer l'enregistrement. Si l'authentification échoue, la fenêtre de la figure 7.30 réapparaît pour recommencer un enregistrement. Si elle réussit, la fenêtre principale montre un message signalant que l'utilisateur est loggué avec une certaine URI comme présenté à la figure 7.31.

FIG. 7.30 – La fenêtre qui apparaît lorsque l'utilisateur doit s'authentifier

Il est possible d'employer le bouton "Add Contact" de la fenêtre principale (voir figure 7.31 pour entrer l'URI SIP d'un autre utilisateur qui apparaitra dans le rectange blanc. En double-cliquant sur cette URI, il est possible de communiquer avec le propriétaire de celle-ci, mais cette partie de l'application n'est pas encore correctement implémentée du fait des développeurs du NIST.

FIG. 7.31 – La fenêtre principale de cette application lorsque l'utilisateur est enregistré